



## Calhoun: The NPS Institutional Archive DSpace Repository

---

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

---

1987-03

# Image texture generation using autoregressive integrated moving average (ARIMA)--mode

Rathmanner, Steven Clifford

---

<http://hdl.handle.net/10945/22277>

---

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

*Downloaded from NPS Archive: Calhoun*



<http://www.nps.edu/library>

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community.

Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School  
411 Dyer Road / 1 University Circle  
Monterey, California USA 93943



DUDLEY KNOX LIBRARY  
NAVAL POSTGRADUATE SCHOOL  
MONTEREY, CALIFORNIA 93943-5002





# NAVAL POSTGRADUATE SCHOOL

## Monterey, California



# THESIS

IMAGE TEXTURE GENERATION USING AUTOREGRESSIVE  
INTEGRATED MOVING AVERAGE (ARIMA) MODELS

by

Steven Clifford Rathmanner

March 1987

Thesis Advisor:

Charles W. Therrien

Approved for public release; distribution is unlimited

T233631



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

## REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b RESTRICTIVE MARKINGS			
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited			
2b DECLASSIFICATION / DOWNGRADING SCHEDULE					
4 PERFORMING ORGANIZATION REPORT NUMBER(S)		5 MONITORING ORGANIZATION REPORT NUMBER(S)			
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL (If applicable) Code 62	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School			
6c ADDRESS (City, State, and ZIP Code)  Monterey, California 93943-5000		7b ADDRESS (City, State, and ZIP Code)  Monterey, California 93943-5000			
8a NAME OF FUNDING/SPONSORING ORGANIZATION	8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER			
8c ADDRESS (City, State, and ZIP Code)		10 SOURCE OF FUNDING NUMBERS			
		PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)  
IMAGE TEXTURE GENERATION USING AUTOREGRESSIVE INTEGRATED MOVING AVERAGE  
(ARIMA) MODELS

12 PERSONAL AUTHOR(S)

Rathmanner, Steven C.

13a TYPE OF REPORT Master's Thesis	13b TIME COVERED FROM _____ TO _____	14 DATE OF REPORT (Year, Month, Day) 1987, March	15 PAGE COUNT 132
---------------------------------------	-----------------------------------------	-----------------------------------------------------	----------------------

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Autoregressive Integrated Moving Average (ARIMA) Stationary; Separable	
FIELD	GROUP	SUB-GROUP		

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis involves investigation of linear filtering models as a means of generating texture in images. Various autoregressive filter models are used to generate various textures, and the results are analyzed to determine relationships between filter parameters and texture characteristics. A two-dimensional counterpart to the autoregressive integrated moving average (ARIMA) model from one-dimensional time series analysis theory is developed and tested for texture modeling applications. All these models are driven by white noise, and to the extent that real images can be reproduced this way, advantages in image texture transmission could be realized. Results of this work indicate that the purely autoregressive models work well for some types of image textures, but that for the textures studied the ARIMA model is not particularly suitable.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. C.W. Therrien			22b TELEPHONE (Include Area Code) (408) 646-3347
			22c OFFICE SYMBOL Code 62T1

Approved for public release; distribution is unlimited

Image Texture Generation Using Autoregressive  
Integrated Moving Average (ARIMA) Models

by

Steven Clifford Rathmanner  
Lieutenant, United States Navy  
B.S., Florida State University, 1979

Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL  
March 1987

## ABSTRACT

This thesis involves investigation of linear filtering models as a means of generating texture in images. Various autoregressive filter models are used to generate various textures, and the results are analyzed to determine relationships between filter parameters and texture characteristics. A two-dimensional counterpart to the autoregressive integrated moving average (ARIMA) model from one-dimensional time series analysis theory is developed and tested for texture modeling applications. All these models are driven by white noise, and to the extent that real images can be reproduced this way, advantages in image texture transmission could be realized. Results of this work indicate that the purely autoregressive models work well for some types of image textures, but that for the textures studied the ARIMA model is not particularly suitable.

5-23-5  
E

## DISCLAIMER

Some terms used in this thesis are registered trademarks of commercial products. Rather than attempt to cite each occurrence of a trademark, all trademarks appearing in this thesis are listed below following the name of the firm holding the trademark:

1. COMTAL, Altadena, California  
Vision One/20
2. Digital Equipment Corporation, Maynard,  
Massachusetts  
VAX/UNIX                  VAX/VMS
3. International Business Machines Corporation,  
Armonk, New York  
IBM System/370 3033

TABLE OF CONTENTS

I.	INTRODUCTION -----	7
II.	THE AUTOREGRESSIVE IMAGE MODEL -----	11
A.	ARBITRARILY SELECTED FILTER COEFFICIENTS; $P = 2, Q = 2$ -----	12
B.	TWO POLE, SEPARABLE AUTOREGRESSIVE MODEL -----	17
C.	FOUR POLE, SEPARABLE AUTOREGRESSIVE MODEL ---	27
1.	Complex Conjugate Poles -----	29
2.	Two Real Poles -----	35
D.	IMAGE TEXTURE ROTATION TRANSFORMATION -----	40
E.	SUMMARY -----	41
III.	IMPLEMENTATION OF AN FIR SUMMATION FILTER IN TWO DIMENSIONS -----	43
IV.	APPLICATION OF THE ARIMA MODEL TO IMAGE TEXTURES -----	56
A.	APPLICATION OF LAPLACIAN INVERSE FILTER TO AUTOREGRESSIVELY GENERATED IMAGES -----	56
B.	AUTOREGRESSIVE FILTER PARAMETER ESTIMATION PROCEDURES -----	60
C.	APPLICATION TO REAL IMAGE TEXTURES -----	61
D.	SUMMARY -----	74
V.	CONCLUSIONS -----	75
APPENDIX A:	COMPUTER PROGRAMS, SUBROUTINES, AND FUNCTIONS -----	77
APPENDIX B:	DERIVATION OF THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE TWO POLE AUTOREGRESSIVE MODEL -----	104

APPENDIX C:	GRAPHICAL RESULTS FOR THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE TWO POLE AUTOREGRESSIVE MODEL -----	106
APPENDIX D:	DERIVATION OF THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE FOUR POLE AUTOREGRESSIVE MODEL -----	109
APPENDIX E:	GRAPHICAL RESULTS FOR THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE FOUR POLE AUTOREGRESSIVE MODEL -----	115
APPENDIX F:	DERIVATION OF THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE FOUR POLE AUTOREGRESSIVE MODEL (WITH TWO POLES ON THE REAL AXIS) -----	120
APPENDIX G:	GRAPHICAL RESULTS FOR THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE FOUR POLE AUTOREGRESSIVE MODEL (WITH TWO POLES ON THE REAL AXIS) -----	124
APPENDIX H:	LAPLACIAN INVERSE FILTER FORMS -----	127
APPENDIX I:	CONVOLUTION OF LAPLACIAN DIFFERENCE OPERATOR AND VARIOUS SIZE FIR INVERSE FILTERS -----	129
LIST OF REFERENCES -----		130
INITIAL DISTRIBUTION LIST -----		131

## I. INTRODUCTION

The purpose of this thesis is to investigate the types and quantity of image textures generated using a two-dimensional (2-D) extension of the Autoregressive Integrated Moving Average (ARIMA) model. For the one-dimensional (e.g., time series) case, the theories and formulas describing this model are outlined in Box and Jenkins [Ref. 1:pp. 85-103]. The 1-D ARIMA model is useful when the time series to be modeled is not stationary but exhibits some homogeneity in the sense that, except for statistical differences between parts of the time series, these different parts of the process behave similarly. In these cases some suitable difference of the process may be stationary, and hence may be accurately modeled by an Autoregressive Moving Average (ARMA) or a purely Autoregressive (AR) model. The resulting stationary time series (generated by an appropriate ARMA or AR filter with white noise input) is applied to an integration or summation filter (the inverse of the difference operation) to generate the original nonstationary time series. [Ref. 1:p. 85] Figure 1-1 shows a block diagram of this process.

This work attempts to extend these concepts to two-dimensional signal processing. In order to simplify the model, the moving average (MA) portion of it will be

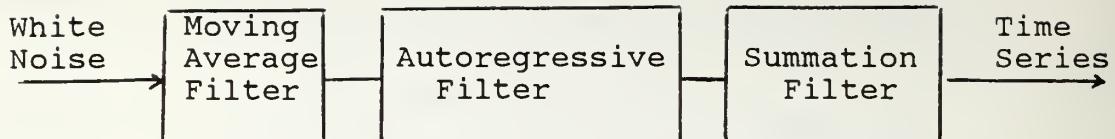


Figure 1-1 Block Diagram for the Autoregressive Integrated Moving Average Model

eliminated (i.e., no zeros in the filter Z transform) so that only purely AR models will be considered for stationary image generation. The procedures for modeling image textures using AR models with white noise input are well established [Ref. 2:pp. 454-456]. However, a suitable two-dimensional difference operation and its inverse must be found to implement the concepts outlined above.

The research is divided into four areas:

- 1) Investigation of the various types of image textures generated using AR models where filter coefficients and size are determined a) arbitrarily, b) using a two-pole separable model, and c) using a four-pole separable model. Separability refers to the fact that the Z transform of the AR filter can be factored into components representing each dimension or direction of the image.
- 2) Selection of a difference operator and a realizable inverse (integration or summation) filter.
- 3) Application of the above autoregressively generated images to the summation filter, and evaluation of these results.
- 4) Attempted reproduction of actual images textures using AR models whose coefficients are determined using the statistics of the image, and comparison of these results to those obtained by a) applying the difference operator to the real image, b) finding the coefficients of the AR model that reproduce the difference image, and c) applying the difference image to the summation filter. This comparison was intended

to discover what improvements, if any, may be realized using the ARIMA model vice a purely ARMA (or AR) model.

The remainder of the thesis is organized as follows. Chapter II contains methods and results of investigating various autoregressive image models. Chapter III deals with the formulation, development, and testing of the two-dimensional summation filter. Chapter IV contains the results of applying various AR-generated images to the summation filter and addresses the application of the ARIMA model to real image textures. Chapter V outlines conclusions on the results and applicability of the ARIMA model. Although the ARIMA modeling was not highly successful in reproducing the textures studied here, plausible reasons are given for their failure and conjectures are made about those circumstances where the model would be more successful. Appendix A provides information on the computer algorithms used to implement the equations governing the above processes. Appendices B through G contain derivations of spectral and autocorrelation equations, and the corresponding graphical results, governing the AR processes in Chapter II. Appendices H and I contain graphical results associated with the inverse filter development in Chapter III.

Image data were generated using computer programs written in FORTRAN, compiled using Version 4.5 under the VAX/VMS Version 4.4 operating system. The images were displayed on the COMTAL Vision One/20. The gray level

intensity range of pixel values is 0 (darkest) to 255 (lightest), so the image data generated had to be scaled to that range for display (see Appendix A).

## II. THE AUTOREGRESSIVE IMAGE MODEL

A two-dimensional signal (such as an image texture) can be modeled using a two-dimensional AR model with white noise input. The governing equations in the spatial domain are of the following form [Ref. 3:pp. 325-326]:

$$y(n,m) = - \sum_{\substack{i=0 \\ (i,j) \neq (0,0)}}^{P-1} \sum_{j=0}^{Q-1} a_{ij} y(n-i,m-j) + w(n,m) \quad (2.1)$$

where  $y(n,m)$  is a signal representing the generated image texture at pixel location  $(n,m)$ ,  $a_{ij}$  is the filter coefficient matrix, and  $w(n,m)$  is a two-dimensional white noise signal. The system function corresponding to the filter of Eq. (2.1) is given by

$$Y(z_1, z_2) = \frac{1}{1 + a_{10}z_1^{-1} + a_{01}z_2^{-1} + a_{11}z_1^{-1}z_2^{-1} + \dots + a_{P-1Q-1}z_1^{-(P-1)}z_2^{-(Q-1)}} \cdot W(z_1, z_2) \quad (2.2)$$

where  $z_1$  and  $z_2$  are the Z transform variables corresponding to spatial coordinates  $n$  and  $m$ . Ideal white noise has an autocorrelation function that is an impulse and a flat (constant) power spectrum with magnitude corresponding to the variance of the white noise process [Ref. 4:pp. 22-26]. Therefore determination of the filter coefficients  $a_{ij}$  will define the generated image process. Procedures will be

outlined later to estimate the coefficients from real image data. At this point analytical methods will be used to select these coefficients and the resulting images will be studied. Figure 2-1 shows an example of a white noise input image.

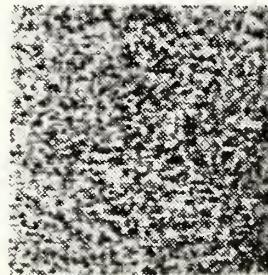


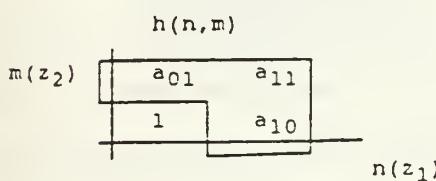
Figure 2-1 White Noise Image

A. ARBITRARILY SELECTED FILTER COEFFICIENTS;  $P = 2$ ,  $Q = 2$

In order to get an initial idea of what types of images might be generated using a  $2 \times 2$  AR filter with white noise input, filter coefficients were at first selected arbitrarily, but subject to a stability constraint. The primary constraint on coefficient selection is that of filter stability. Using the DeCarlo-Strintzis Theorem dealing with multidimensional filter stability [Ref. 3:pp. 197-198], alternately setting  $z_1 = 1$  and  $z_2 = 1$  and determining the location of the pole in the remaining dimension will indicate whether or not the filter is stable. If the

magnitude of the pole in the remaining dimension is less than 1, the filter is stable. Even with this condition, however, there are an infinite number of possible filter coefficient combinations. The additional constraint of  $a_{10} = a_{01}$  can be used, and comparisons of results using various values of  $a_{11}$  can be made.

Figure 2-2 shows the form and directionality convention used for the autoregressive filter, along with the corresponding difference equation and its Z transform.



$$h(n,m) = \begin{bmatrix} a_{01} & a_{11} \\ 1 & a_{10} \end{bmatrix}$$

$$m(z_2)$$

$$n(z_1)$$

$$y(n,m) = - \sum_{i=0}^1 \sum_{j=0}^1 a_{ij} \cdot y(n-i,m-j) + w(n,m)$$

$$(i,j) \neq (0,0)$$

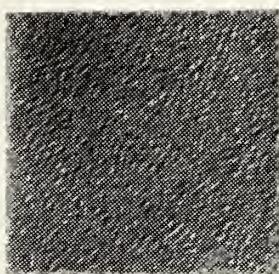
$$Y(z_1, z_2) = H(z_1, z_2) \cdot W(z_1, z_2) = \frac{1}{1 + a_{10}z_1^{-1} + a_{01}z_2^{-1} + a_{11}z_1^{-1}z_2^{-1}} \cdot W(z_1, z_2)$$

Figure 2-2 Autoregressive Filter Impulse Response, Difference Equation, and Z Transform

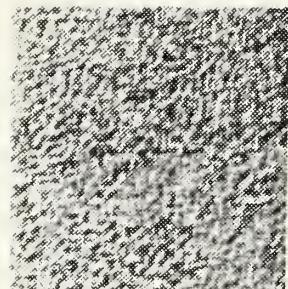
Although it is difficult to make precise predictions in two dimensions, one can expect that the sign and magnitude of  $a_{10}$  or  $a_{01}$  would influence the correlation between pixels

in the corresponding directions. For example, a positive value for  $a_{10}$  might be expected to yield an image with substantial variation in the n direction (low correlation), particularly if the magnitude of  $a_{10}$  is near 1. A negative value for  $a_{01}$  with magnitude near 1 might yield an image with lower variations in pixel intensity (high correlation) in the m direction. Since the filter is not necessarily separable (i.e., the denominator of  $H(z_1, z_2)$  cannot be factored into the form  $D(z_1) \cdot D(z_2)$ ), conclusions drawn from this line of reasoning may not be completely correct.

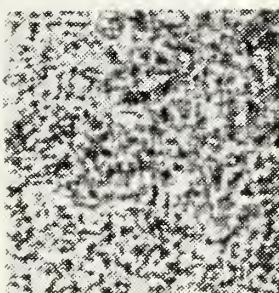
Initial attempts at generating images with arbitrarily selected coefficients yielded rather uninteresting results having very little contrast or discernible pattern. Continued experimentation with combinations where  $a_{10} = a_{01} < 0$  eventually yielded more interesting image textures. Figures 2-3 and 2-4 show the results of using the constraint  $a_{10} = a_{01} = -0.35$  and various values of  $a_{11}$  for the filter coefficients. For positive values of  $a_{11}$ , the images are rather "grainy," with higher magnitudes yielding a somewhat "finer" graininess. There are also some overall intensity differences observed. For the negative values of  $a_{11}$ , the results are much more interesting. As the magnitude increases, there is a gradually more noticeable upper left to lower right orientation of the image texture, and the variations from lower left to upper right become smoother as well. Using  $a_{10} = a_{01} = -0.38$  and  $a_{11} = -0.24$



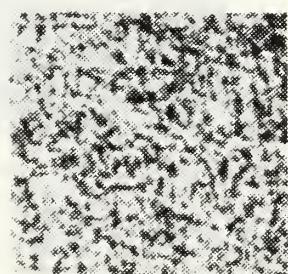
$a_{10} = a_{01} = -0.35$   
 $a_{11} = 0.8$



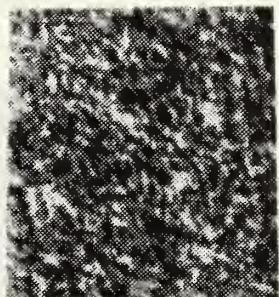
$a_{10} = a_{01} = -0.35$   
 $a_{11} = 0.5$



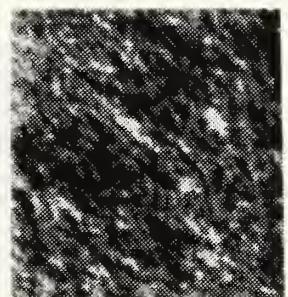
$a_{10} = a_{01} = -0.35$   
 $a_{11} = 0.2$



$a_{10} = a_{01} = -0.35$   
 $a_{11} = 0.0$

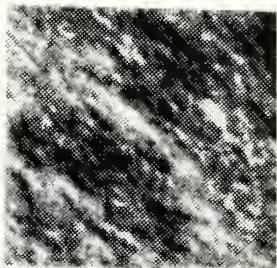


$a_{11} = a_{01} = -0.35$   
 $a_{11} = -0.1$



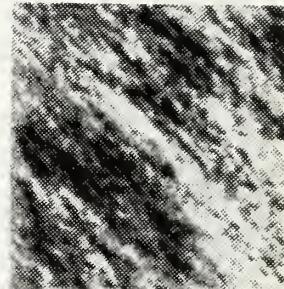
$a_{10} = a_{01} = -0.35$   
 $a_{11} = -0.2$

Figure 2-3 Images Generated Using Arbitrarily Selected Filter Coefficients



$$a_{10} = a_{01} = -0.35$$

$$a_{11} = -0.25$$



$$a_{10} = a_{01} = -0.35$$

$$a_{11} = -0.27$$



$$a_{10} = a_{01} = -0.35$$

$$a_{11} = -0.3$$



$$a_{10} = a_{01} = -0.38$$

$$a_{11} = -0.24$$

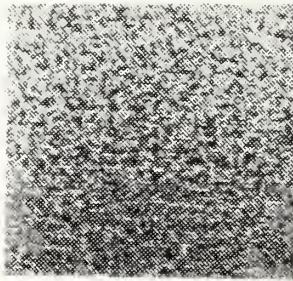
Figure 2-4 Images Generated Using Arbitrarily Selected Filter Coefficients

yields minor variations in texture pattern and overall image intensity when compared to the previous case.

Using  $a_{10} = a_{01} = 0.35$  and varying  $a_{11}$  from 0.0 to 0.8 (Figure 2-5), the images obtained deviate very little from the mean intensity value, and possess minor differences in graininess. With these positive coefficients some negative correlation might be expected, and the fact that these images are "grainy" indicates the existence of some negative correlation or high spatial frequency characteristics. On initial examination, however, the low contrast of the generated images tends to obscure the observed graininess.

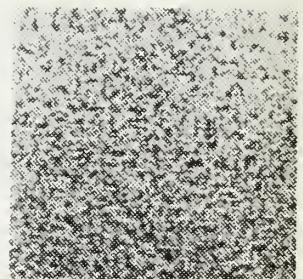
#### B. TWO POLE, SEPARABLE AUTOREGRESSIVE MODEL

In general, it is difficult to relate the nature or properties of a two-dimensional filter to the precise nature of an image texture that may be generated when white noise is applied to that filter. In order to simplify the effort and to obtain a better understanding of the problem, the case where the filter (and resulting image texture) are separable is considered. For the two pole separable case considered here, the filter transfer function can be factored into expressions in  $z_1$  alone and  $z_2$  alone. The expressions in  $z_1$  and  $z_2$  each have one pole on the real axis in their respective Z domains. Figure 2-6 illustrates the filter structure, the corresponding difference equation, and its Z transform.



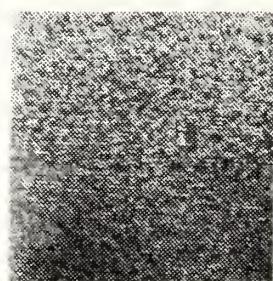
$a_{10} = a_{01} = 0.35$

$a_{11} = 0.0$



$a_{10} = a_{01} = 0.35$

$a_{11} = 0.2$



$a_{10} = a_{01} = 0.30$

$a_{11} = 0.5$



$a_{10} = a_{01} = 0.35$

$a_{11} = 0.8$

Figure 2-5 Images Generated Using Arbitrarily Selected Filter Coefficients

$$y(n,m) = - \sum_{i=0}^1 \sum_{j=0}^1 a_{ij} \cdot y(n-i, m-j) + w(n,m)$$

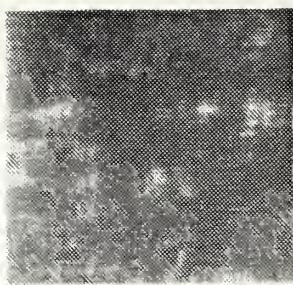
$$(i,j) \neq (0,0)$$

$(a_{11} = a_{10} \cdot a_{01})$

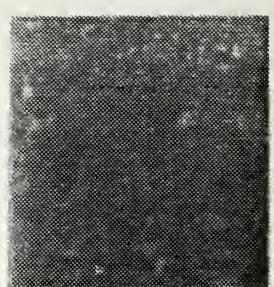
$$\begin{aligned} Y(z_1, z_2) &= H(z_1, z_2) \cdot W(z_1, z_2) = \frac{1}{1+a_{10}z_1^{-1}} \cdot \frac{1}{1+a_{01}z_2^{-1}} \cdot W(z_1, z_2) \\ &= \frac{1}{1+a_{10}z_1^{-1}+a_{01}z_2^{-1}+a_{10} \cdot a_{01}z_1^{-1}z_2^{-1}} \cdot W(z_1, z_2) \end{aligned}$$

Figure 2-6 Autoregressive Filter Form, Difference Equation, and Z Transform

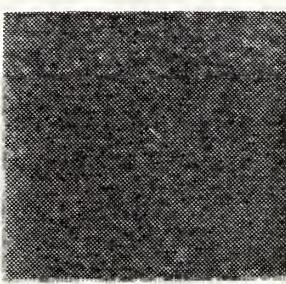
Here it is relatively easy to relate stability of the filter to the location of the poles in the  $z_1$  and  $z_2$  planes. Since the quarter plane filter is separable and the components are causal, one-dimensional filter stability theory can be used to state that the poles in each plane must have magnitude less than 1 to ensure filter stability. Figures 2-7 through 2-9 show images resulting from this model for various values of  $a_{10}$  and  $a_{01}$ . Note that the sign convention used for the filter difference equation and z transform results in poles on the negative side of the real axis for positive values of  $a_{10}$  or  $a_{01}$ , and vice versa.



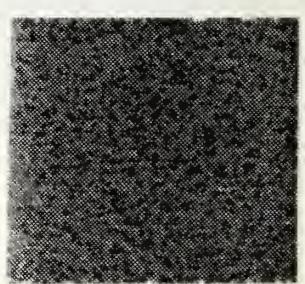
$a_{10} = a_{01} = 0.95$



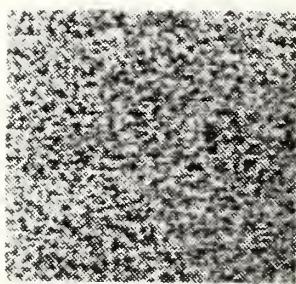
$a_{10} = a_{01} = 0.8$



$a_{10} = a_{01} = 0.5$



$a_{10} = a_{01} = 0.25$

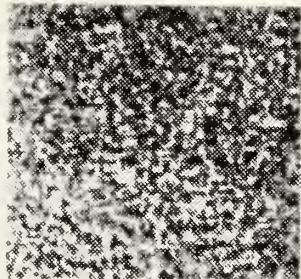


$a_{10} = a_{01} = 0.1$



$a_{10} = a_{01} = -0.95$

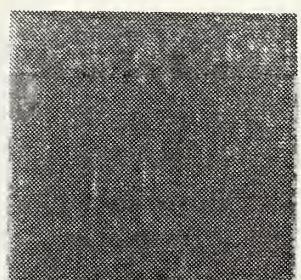
Figure 2-7 Images Generated Using a Two-Pole Autoregressive Model



$$a_{10} = a_{01} = -0.25$$



$$a_{10} = 0.95 \quad a_{01} = 0.5$$



$$a_{10} = 0.5 \quad a_{01} = 0.95$$



$$a_{10} = 0.95 \quad a_{01} = 0.75$$

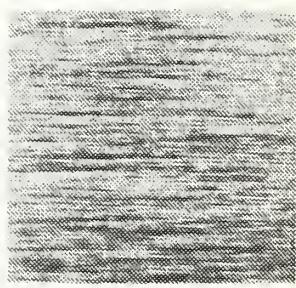


$$a_{10} = -0.95 \quad a_{01} = 0.25$$

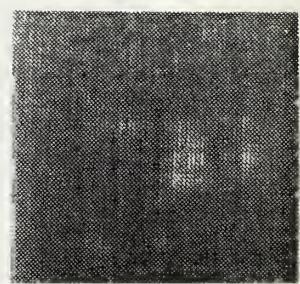


$$a_{10} = -0.95 \quad a_{01} = -0.25$$

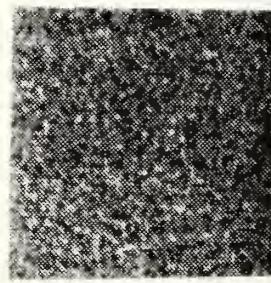
Figure 2-8 Images Generated Using a Two-Pole Autoregressive Model



$a_{10} = -0.95 \quad a_{01} = 0.75$



$a_{10} = 0.95 \quad a_{01} = -0.95$



$a_{10} = 0.25 \quad a_{01} = -0.25$

Figure 2-9 Images Generated Using a Two-Pole Autoregressive Model

Careful comparison of the images resulting from various combinations of  $a_{10}$  and  $a_{01}$  leads to the following observations:

- 1) When poles are located in the same place in the  $z_1$  and  $z_2$  planes on the negative side of the real axis, magnitudes near 1 yield a fine graininess with patchy areas and low overall contrast. As the magnitude of the pole decreases, the graininess becomes more coarse and the result is more like the original white noise input. No directional quality in the image pattern is observed.
- 2) When poles are located in the same place on the positive side of the  $z_1$  and  $z_2$  real axes, a somewhat different result is observed. For magnitudes near 1, an image of patchy light and dark areas results, with differing amounts of correlation between pixels in different areas. Slightly discernible "lines" in both the horizontal and vertical directions are also observed. For lower pole magnitudes on the positive side of the real axis, the decreased effect of the filter on the white noise input is again observed. This result is more like the white noise and has more contrast than the corresponding result using poles on the negative side of the real axis.
- 3) For pole placements in the  $z_1$  and  $z_2$  planes which are on the negative side of the real axis and are of unequal magnitude, the results have a very fine graininess and low contrast. Some slight directionality is observable in the image patterns, with lower frequency variations evident in the direction corresponding to the pole with smaller magnitude.
- 4) For pole placements in the  $z_1$  and  $z_2$  planes which are on the positive side of the real axis and are of unequal magnitude, much more directionality and variation is observable in the image pattern.
- 5) As the poles are placed on opposite sides of the real axis and are separated by a greater distance, directionality becomes more evident (with higher frequency variations in the direction of the more negative pole). As the pole separation becomes greater and as the pole magnitudes become closer to 1, smoother sinusoidal variations are evident.
- 6) When the pole magnitudes are equal and have opposite sign, the image generated using pole magnitudes close

to 1 exhibits high frequency sinusoidal variations in the direction of the negative pole. The image generated with the lower magnitude poles, as would be expected from the above results, resembled the unfiltered white noise.

In order to explain these image patterns analytically, analysis of the power spectrum and autocorrelation function of this process is useful. Since this model is separable, the analysis can be conducted in each direction separately. The power spectrum is defined by [Ref. 4:pp. 24-34]:

$$S_Y(\omega) = \sigma^2 |H(e^{j\omega})|^2 = \sigma^2 H(e^{j\omega}) H(e^{-j\omega}) \quad (2.3a)$$

where

$$H(e^{j\omega}) = H(z) |_{z=e^{j\omega}} \quad (2.3b)$$

and for this case

$$H(z) = \frac{1}{1 + \alpha z} \quad (2.3c)$$

Here  $\sigma^2$  is the magnitude of the white noise power spectrum. We can assume that  $\sigma^2 = 1$  with no loss of generality of the results, since  $\sigma^2$  does not affect the shape of the frequency response.

The autocorrelation function is related to the filter transfer function through the equations [Ref. 5:pp. 391-395]:

$$h(n) = z^{-1}[H(z)] \quad (2.4a)$$

$$y(n) = h(n) * w(n) \quad (2.4b)$$

$$R_y(\ell) = \sigma^2 \sum_{n=-\infty}^{\infty} h(n) \cdot h(n-\ell) \quad (2.4c)$$

Specific forms of the power spectrum and autocorrelation function are given in Appendix B. Since  $R_y(\ell) = R_y(-\ell)$  [Ref. 5:p. 388], calculating the expression for  $R_y(\ell)$ ,  $\ell < 0$  is not necessary. From Appendix B, the results are:

$$S_y(\omega) = \frac{1}{1 + 2\alpha \cos(\omega) + \alpha^2} \quad (2.5)$$

$$R_y(\ell) = \frac{(-\alpha)^{\ell}}{1 - \alpha^2} \quad \ell \geq 0 \quad (2.6)$$

Appendix C shows the results of these equations graphically for various values of  $\alpha$ . The relationship between the power spectra and their corresponding autocorrelation functions conforms to the expected results from theory (i.e., low frequency spectrum with smooth autocorrelation function, and high frequency power spectrum with rapidly varying autocorrelation function) [Ref. 6:pp. 139-142]. The plots in Appendix C also demonstrate that: 1) For poles on the positive side of the real axis in the Z plane low frequencies predominate and for poles on the negative side of the real axis high frequencies predominate, and 2) Lower

magnitudes of  $\alpha$  result in a broader power spectrum and a wider range of frequencies of significant magnitude. Both of these observations agree with the image results. For images generated using a more negative pole in a given direction, fine, high frequency graininess is observed in that direction (though the low contrast or low variation about the mean intensity may tend to make this effect less noticeable). When a more positive pole is used, lower frequency variations are more evident in the corresponding direction. As lower magnitudes are used for  $\alpha$  in a given direction, more random variations (indicative of a wider range of significant frequency components) are observed in that direction. The form of the autocorrelation function for these cases approaches the autocorrelation function for white noise, i.e., an impulse. Negative poles should yield high frequencies since the negative side of the real axis in the Z plane represents a digital spatial frequency of  $\pi$ , while positive poles in the Z plane correspond to a digital spatial frequency of zero. Note that even when the poles are placed such that a spatial frequency of zero should predominate, there are some low frequency random variations in the resulting images. Since the power spectra of the positive poles all contain some non-zero frequency components (they are not perfect impulses at zero), this characteristic is expected.

### C. FOUR POLE, SEPARABLE AUTOREGRESSIVE MODEL

For the cases considered in this section,  $H(z)$  again can be factored into expressions in  $z_1$  and  $z_2$ . However here each factor is a 2nd degree polynomial with two poles in the denominator. Figure 2-10 illustrates the filter structure, the applicable difference equation, and the corresponding z transform.

$$h(n, m)$$

$$y(n, m) = - \sum_{i=0}^2 \sum_{j=0}^2 a_{ij} \cdot y(n-i, m-j) + w(n, m)$$

$$(i, j) \neq (0, 0)$$

$$Y(z_1, z_2) = H(z_1, z_2) \cdot W(z_1, z_2) = \frac{1}{1+a_{10}z_1^{-1}+a_{20}z_1^{-2}} \cdot \frac{1}{1+a_{01}z_2^{-1}+a_{02}z_2^{-2}} \cdot W(z_1, z_2)$$

$$= \frac{1}{1+a_{10}z_1^{-1}+a_{20}z_1^{-2}+a_{01}z_2^{-1}+a_{02}z_2^{-2}+a_{11}z_1^{-1}z_2^{-1}+a_{21}z_1^{-2}z_2^{-1}+a_{12}z_1^{-1}z_2^{-2}+a_{22}z_1^{-2}z_2^{-2}} \cdot W(z_1, z_2)$$

where

$$a_{11} = a_{10} \cdot a_{01}; \quad a_{21} = a_{10} \cdot a_{02}; \quad a_{12} = a_{02} \cdot a_{10}; \quad a_{22} = a_{20} \cdot a_{02}$$

Figure 2-10 Autoregressive Filter Form, Difference Equation, and Z Transform

Since all of the  $a_{ij}$  coefficients are real, the poles must 1) both be on the real axis, or 2) occur in complex conjugate pairs in the  $z_1$  and  $z_2$  planes. Again, pole magnitudes must be less than 1 to ensure filter stability. We will assume here that the poles in each of the factors have equal magnitudes and opposite (or 0 or  $\pm\pi$ ) phase. Letting

$$\alpha_1 = \text{magnitude of poles in the } z_1 \text{ plane}$$

$$\theta_1 = \text{pole angle (phase) in the } z_1 \text{ plane}$$

$$\alpha_2 = \text{magnitude of poles in the } z_2 \text{ plane}$$

$$\theta_2 = \text{pole angle (phase) in the } z_2 \text{ plane}$$

and using Euler's relation, the denominators of  $H(z_1)$  and  $H(z_2)$  can be expressed as follows:

$$1+a_{11}z_1^{-1}+a_{20}z_1^{-2} = (1-\alpha_1 e^{j\theta_1} z_1^{-1})(1-\alpha_1 e^{-j\theta_1} z_1^{-1}) = 1-2\alpha_1 \cos(\theta_1) \cdot z_1^{-1} + \alpha_1^2 z_1^{-2}$$

$$1+a_{01}z_2^{-1}+a_{02}z_2^{-2} = (1-\alpha_2 e^{j\theta_2} z_2^{-1})(1-\alpha_2 e^{-j\theta_2} z_2^{-1}) = 1-2\alpha_2 \cos(\theta_2) z_2^{-1} + \alpha_2^2 z_2^{-2}$$

Hence:

$$a_{10} = -2\alpha_1 \cos(\theta_1)$$

$$a_{01} = -2\alpha_2 \cos(\theta_2)$$

$$a_{20} = \alpha_1^2$$

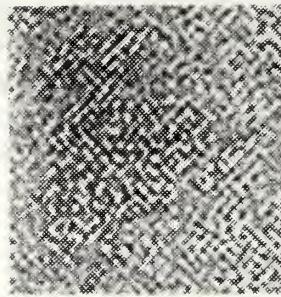
$$a_{02} = \alpha_2^2$$

This gives a relationship between the pole magnitude and angle in the  $Z$  domain and the filter coefficients in the spatial domain.

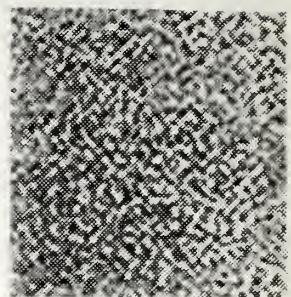
## 1. Complex Conjugate Poles

Figures 2-11 through 2-13 illustrate images generated using this model for various complex conjugate pole combinations in the  $z_1$  and  $z_2$  directions. In comparing each of these image textures in terms of the relative effect of pole positioning in each direction, the following observations can be made:

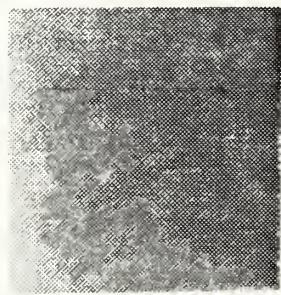
- 1) For images generated using poles of equal magnitude and angle in both directions, graininess with no directionality to the pattern resulted. Higher pole angles yielded finer (higher frequency) graininess and less contrast. Lower magnitude poles yielded a more random and less structured graininess pattern at the same pole angle.
- 2) Using a pole angle of zero (pole on positive real axis) in one direction and a pole of some non-zero angle in the other direction yielded images with highly directional sinusoidal patterns. The direction corresponding to the pole on the real axis was not totally devoid of variation, but variations were slow, i.e., of very low frequency. The spatial frequency of the sinusoidal pattern can be increased by increasing the pole angle. Large magnitude, high pole angle combinations yielded much cleaner and more structured textures than low magnitude, low pole angle combinations. Lower magnitude, high pole angle combinations yielded less structured textures where directionality was evident but the sinusoidal pattern was obscured. Low magnitude, low pole angle combinations yielded very random, non-structured textures of relatively high contrast.
- 3) Using poles in the  $z_1$  and  $z_2$  planes with the same magnitude but different pole angles resulted in some directionality if there was a sufficiently large magnitude and difference in the pole angles. As observed earlier, the direction with the higher pole angle had the higher spatial frequency. Large pole magnitudes (close to 1) resulted in more structured but rather low-contrast images (the low contrast seemed to obscure the high frequency nature of the pattern somewhat). Pole angles in  $z_1$  and  $z_2$  that were of close value made it difficult to detect the higher frequency (higher pole angle) direction. Low



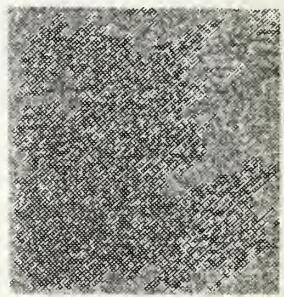
$$\begin{aligned}z_1 &= 0.8e^{\pm j\frac{\pi}{3}} \\z_2 &= 0.8e^{\pm j\frac{\pi}{3}}\end{aligned}$$



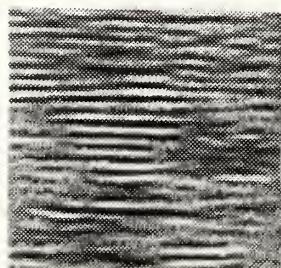
$$\begin{aligned}z_1 &= 0.7e^{\pm j\frac{\pi}{3}} \\z_2 &= 0.7e^{\pm j\frac{\pi}{3}}\end{aligned}$$



$$\begin{aligned}z_1 &= 0.9e^{\pm j\frac{2\pi}{3}} \\z_2 &= 0.9e^{\pm j\frac{2\pi}{3}}\end{aligned}$$



$$\begin{aligned}z_1 &= 0.6e^{\pm j\frac{2\pi}{3}} \\z_2 &= 0.6e^{\pm j\frac{2\pi}{3}}\end{aligned}$$

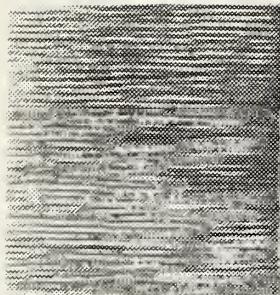


$$\begin{aligned}z_1 &= 0.9e^{\pm j0} \\z_2 &= 0.9e^{\pm j\frac{\pi}{3}}\end{aligned}$$

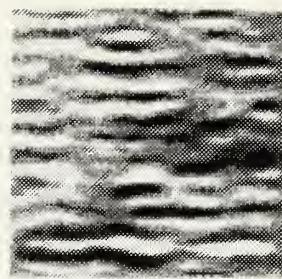


$$\begin{aligned}z_1 &= 0.9e^{\pm j\frac{\pi}{3}} \\z_2 &= 0.9e^{\pm j0}\end{aligned}$$

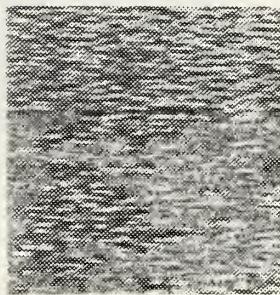
Figure 2-11 Images Generated Using a Four Pole Autoregressive Model (Poles Listed Below Image)



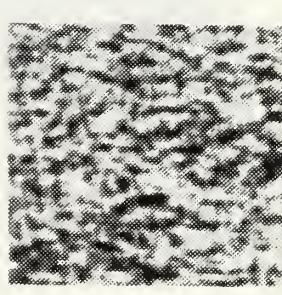
$$\begin{aligned}z_1 &= 0.9e^{\pm j0} \\z_2 &= 0.9e^{\pm j\frac{2\pi}{3}}\end{aligned}$$



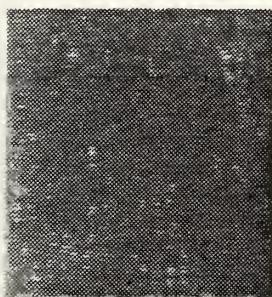
$$\begin{aligned}z_1 &= 0.9e^{\pm j0} \\z_2 &= 0.9e^{\pm j\frac{\pi}{6}}\end{aligned}$$



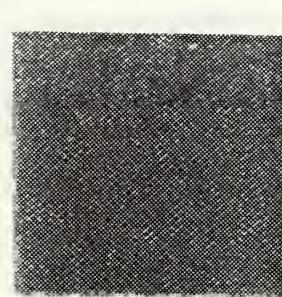
$$\begin{aligned}z_1 &= 0.6e^{\pm j0} \\z_2 &= 0.6e^{\pm j\frac{2\pi}{3}}\end{aligned}$$



$$\begin{aligned}z_1 &= 0.6e^{\pm j0} \\z_2 &= 0.6e^{\pm j\frac{\pi}{6}}\end{aligned}$$

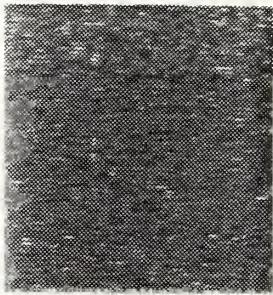


$$\begin{aligned}z_1 &= 0.9e^{\pm j\frac{\pi}{6}} \\z_2 &= 0.9e^{\pm j\frac{5\pi}{6}}\end{aligned}$$

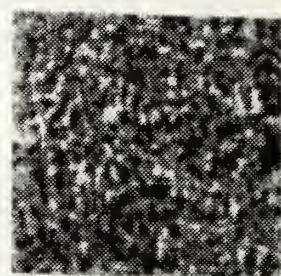


$$\begin{aligned}z_1 &= 0.9e^{\pm j\frac{5\pi}{12}} \\z_2 &= 0.9e^{\pm j\frac{7\pi}{12}}\end{aligned}$$

Figure 2-12 Images Generated Using a Four Pole Autoregressive Model (Poles Listed Below Image)



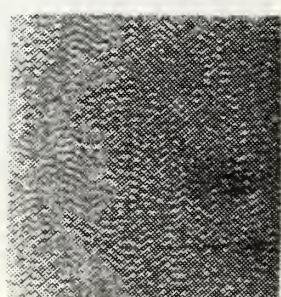
$$\begin{aligned} z_1 & \quad 0.6e^{\pm j\frac{\pi}{6}} \\ z_2 & \quad 0.6e^{\pm j\frac{5\pi}{6}} \end{aligned}$$



$$\begin{aligned} z_1 & \quad 0.5e^{\pm j\frac{\pi}{6}} \\ z_2 & \quad 0.5e^{\pm j\frac{\pi}{10}} \end{aligned}$$



$$\begin{aligned} z_1 & \quad 0.9e^{\pm j\frac{\pi}{6}} \\ z_2 & \quad 0.8e^{\pm j\frac{\pi}{2}} \end{aligned}$$



$$\begin{aligned} z_1 & \quad 0.6e^{\pm j\frac{5\pi}{12}} \\ z_2 & \quad 0.9e^{\pm j\frac{7\pi}{12}} \end{aligned}$$

Figure 2-13 Images Generated Using a Four Pole Autoregressive Model (Poles Listed Below Image)

magnitudes basically negated the pole angle effects and yielded a very random, unstructured, high contrast texture.

- 4) Where pole magnitudes were close and pole angles were different in the  $z_1$  and  $z_2$  planes, some directionality in the texture was observed. Again, the high pole angle direction yielded the highest frequency. When the pole angles in both Z domains had similar values and the magnitudes of the poles differed, graininess with little or no discernible directionality or structure resulted.

These observations are consistent with classical pole-zero frequency response analysis [Ref. 7:pp. 323-331]. There is a direct relationship between pole angle and spatial frequency in a given direction, and the magnitude of the poles affects the amount of structure and definition of the sinusoidal pattern of a given frequency in a given direction. Higher magnitude poles result in a narrower bandwidth of the filter and yield more structure and sinusoidal pattern definition. Low pole magnitudes give the filter wider bandwidth and yield images with less structure and definition and more randomness in a given direction. While directional dependencies are evident given pole magnitude and angle in a given direction, it does not appear that a pattern in one direction is totally independent of a pattern in the other direction. This would be expected, even though the model is separable, due to the cross terms in the filter structure.

Filter power spectrum and autocorrelation analysis can be conducted in this case, as in the case of the two pole model. The derivations for  $S_y(\omega)$  and  $R_y(\ell)$  are

somewhat more involved, and are given in Appendix D. The resulting expressions for  $S_Y(\omega)$  and  $R_Y(l)$  from Appendix D are:

$$S_Y(\omega) = \frac{1}{1+\alpha^4 + 2(\alpha^2 - 2[\alpha^3 - \alpha]\cos(\theta)\cos(\omega) + \alpha^2(\cos(2\theta) + \cos(2\omega)))} \quad (2.7)$$

$$R_Y(l) = \frac{\alpha^2}{2\sin^2\theta} \frac{(\cos(l\theta))}{1-\alpha^2} - \frac{\cos((2+l)\theta) - \alpha^2\cos(l\theta)}{1+\alpha^4 - 2\alpha^2\cos(2\theta)} \quad (l \geq 0) \quad (2.8)$$

The plots of these functions for the various pole magnitudes and angles used are given in Appendix E. The  $\theta = 0$  case is equivalent to having 2 poles on the real axis at a given magnitude in the Z plane. As would be expected, the power spectrum for each model showed higher magnitudes at digital frequencies close to the pole angle. Higher pole magnitudes yielded sharper, more well-defined power spectrum magnitudes at the given frequency, and lower pole magnitudes yielded less well defined more spread-out power spectra. Low pole magnitudes almost completely obliterated evidence of low frequency power spectrum components, and degraded its definition and sharpness at higher frequencies. This corresponds to the observed results in the image textures generated. The autocorrelation functions also reflect the appropriate relationship to the power spectra as outlined in the discussion for the two pole case, i.e., greater

variation in the autocorrelation function indicates greater variation between pixels a given distance apart, which in turn implies higher spatial frequencies.

## 2. Two Real Poles

Rather than using complex conjugate pole locations to obtain real filter coefficient values, two poles on the real axis may also be used for a given direction. They may be placed at different locations on the real axis, or they may be placed together. The latter situation is equivalent to the  $\theta = 0$  (or  $\theta = \pm\pi$  if placed on the negative real axis) case, as mentioned above. For the two real pole case, the relevant equations are:

$$\begin{aligned} H(z) &= \frac{1}{(1-\alpha_a z^{-1})(1-\alpha_b z^{-1})} = \frac{1}{1-(\alpha_a + \alpha_b)z^{-1} + \alpha_a \alpha_b z^{-2}} \\ &= \frac{1}{1+a_{01}z^{-1}+a_{02}z^{-2}} \end{aligned}$$

where

$$a_{01} = -(\alpha_a + \alpha_b)$$

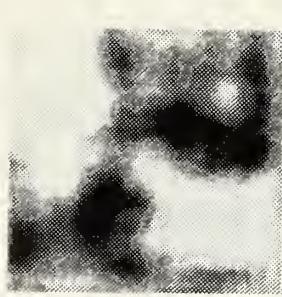
$$a_{02} = \alpha_a \cdot \alpha_b$$

For these experiments a transfer function of the complex conjugate pole form was used for the  $z_1$  direction, with  $\alpha_1 = 0.9$  and  $\theta_1 = 0$ . For the  $z_2$  direction a transfer function with two poles on the real axis was used. The

image textures that result for various values of  $\alpha_a$  and  $\alpha_b$  are given in Figures 2-14 and 2-15. Some observations can be made about these results:

- 1) With poles placed at the same value on the  $z_2$  real axis, rather unstructured, low frequency variations are observed in the  $z_2$  direction. The more positive poles result in very slow variation in the image texture, while the lower magnitude positive poles show more variation in the  $z_2$  direction. When the poles are placed in the same location on the negative side of the real axis, a low contrast image with some noticeable high frequency variations results.
- 2) As the poles are moved farther apart on the  $z_2$  real axis, high frequency variations with increasing structure and oscillatory form are evidenced in the  $z_2$  direction.
- 3) When poles with equal magnitude and opposite sign are used, fairly structured high frequency variations are evidenced in certain areas of the image, while low frequency variations are evident in other areas in the  $z_2$  direction. Higher magnitude poles yield more discernible, structured variations, while lower magnitude poles of opposite sign yield discernible but non-oscillatory high frequency variations in certain areas of the image.

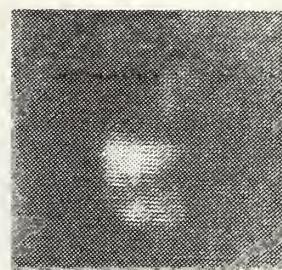
Of particular interest is the fact that two poles placed at the same value on the negative real axis in the  $z_2$  plane yielded some high frequency variations. This is in keeping with the fact that values on the negative real axis correspond to a pole angle (and corresponding digital frequency) of  $\theta = \pi$ . The presence of poles on the negative side of the real axis of the  $z_2$  plane seems to give rise to the high frequency variations with gradually more structure and oscillatory appearance as the pole is moved to the left (more negative).



$\alpha_a = 0.9 \quad \alpha_b = 0.9$



$\alpha_a = 0.5 \quad \alpha_b = 0.5$



$\alpha_a = -0.9 \quad \alpha_b = -0.9$



$\alpha_a = 0.9 \quad \alpha_b = 0.5$



$\alpha_a = 0.9 \quad \alpha_b = 0.0$



$\alpha_a = 0.9 \quad \alpha_b = -0.2$

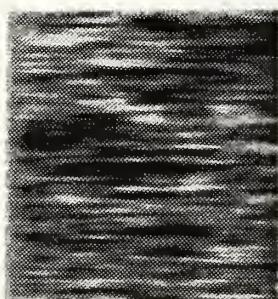
Figure 2-14 Images Generated Using a Four Pole  
(Two Real Poles) Autoregressive Model



$\alpha_a = 0.8 \quad \alpha_b = -0.9$



$\alpha_a = 0.9 \quad \alpha_b = -0.9$



$\alpha_a = 0.5 \quad \alpha_b = -0.5$

Figure 2-15 Images Generated Using a Four Pole  
(Two Real Poles) Autoregressive Model

The expressions for the power spectrum and autocorrelation function for the random process produced by driving the filter of Figure 2-10 with white noise are derived in Appendix F. It is shown there that the power spectral density and the autocorrelation function are given by:

$$S_Y(\omega) = \frac{1}{1 - 2(\alpha_a^2 + \alpha_b^2 + \alpha_a^2\alpha_b^2)\cos(\omega) + 2\alpha_a\alpha_b\cos(2\omega) + \alpha_a^2 + 2\alpha_a^2\alpha_b^2 + \alpha_b^2} \quad (2.9)$$

$$R_Y(\ell) = \frac{1}{(\alpha_a - \alpha_b)^2} \left[ \frac{\alpha_a^{2+\ell}}{1 - \alpha_a^2} - \frac{\alpha_a^{\ell+1}\alpha_b^{\ell+1}}{1 - \alpha_a\alpha_b} + \frac{\alpha_b^{2+\ell}}{1 - \alpha_b^2} \right] \quad (2.10)$$

Plots of these functions for the various values used in this section are given in Appendix G. The power spectrum results are consistent with the observed image spatial frequency characteristics. Both low and high frequency components were contained in some of the power spectra, and were manifested in the corresponding images as both low and high frequency variations in the  $z_2$  direction. The nature of the autocorrelation functions related to the power spectra that contained low and high frequency components was interesting. Autocorrelation functions with much variation but all positive values, rather than the equal magnitude

positive and negative values evidenced in earlier results, seems to reflect the higher level of correlation related to the low frequency (smoother variations) aspect of the image texture variations.

#### D. IMAGE TEXTURE ROTATION TRANSFORMATION

If an image signal  $x(n_1, n_2)$  consists of a rotated version of another image  $w(m_1, m_2)$  such that  $n_1 = Im_1 + Jm_2$  and  $n_2 = Km_1 + Lm_2$ , where  $I, J, K$ , and  $L$  are integers and  $IL - JK \neq 0$ , then the Z transform  $X(z_1, z_2)$  is given by  $W(z_1^I, z_2^K, z_1^J, z_2^L)$  [Ref. 3:p. 182]. A  $45^\circ$  rotation corresponds to  $I = 1, K = 1, J = 1, L = -1$ . If we use the four pole separable result for  $H(z_1, z_2)$ , as shown in Figure 2-10, and apply the above transformation  $(z_1 \rightarrow z_1^1 z_2^1, z_2 \rightarrow z_1^1 z_2^{-1})$ , we find after simplification:

$$H_R(z_1 z_2) = \frac{1}{1+a_{01}z_1^{-1}z_2^{-1}+a_{02}z_1^{-2}z_2^2+a_{10}z_1^{-1}z_2^{-1}+a_{10}a_{01}z_1^{-2}+a_{10}a_{02}z_1^{-3}z_2^{-1}+a_{20}z_1^{-2}z_2^{-2}+a_{20}a_{01}z_1^{-3}z_2^{-1}+a_{20}a_{02}z_1^{-4}} \quad (2.11)$$

Notice that this transfer function is not separable but consists of a rotated version of a separable filter. Figure 2-16 illustrates the support of the denominator polynomial for this filter. It has the form of a non-symmetric half-plane infinite impulse response (IIR) filter, so it is recursively computable.

0	0	$a_{20}$	0	
0	$a_{20}a_{01}$	0	$a_{10}$	
$a_{20}a_{02}$	0	$a_{10}a_{01}$	0	1
0	$a_{10}a_{02}$	0	$a_{01}$	0
0	0	$a_{02}$	0	0

Figure 2-16 Rotation Transformation Filter Form

The application of this filter, using filter coefficients of the four pole separable filter with poles at  $z_1 \rightarrow 0.9e^{\pm j\frac{\pi}{2}}$ ,  $z_2 \rightarrow 0.9e^{\pm j0}$  in the original separable filter yielded the result shown in Figure 2-17.



Figure 2-17 Result of Rotation Transformation

#### E. SUMMARY

Autoregressive models can produce a variety of image textures. For general two-dimensional models, the system functions are generally not factorable and singularities

occur on surfaces, not at isolated points. For these reasons it is difficult to design two-dimensional filters for images and predict the resulting character of the images. Indeed, even to ensure stability of the filter is not trivial. As a result we concentrated here on separable forms, which by their nature are much easier to analyze. Certain types of texture patterns using various separable autoregressive models can be predicted based on filter pole placement in the  $z_1$  and  $z_2$  planes. Arbitrary or random selection of filter coefficients can yield interesting but generally unpredictable results. Obviously, an infinite number of variations on the models above could be attempted. Ultimately, the anticipated utility of the textures generated will guide the process of model and parameter selection.

### III. IMPLEMENTATION OF AN FIR SUMMATION FILTER IN TWO DIMENSIONS

To implement the 2-D ARIMA model, the inverse of a filter representing a suitable difference operator is needed. One possible 2-D difference operator is the Laplacian, which has the impulse response shown in Figure 3-1 [Ref. 8:pp. 212-213].

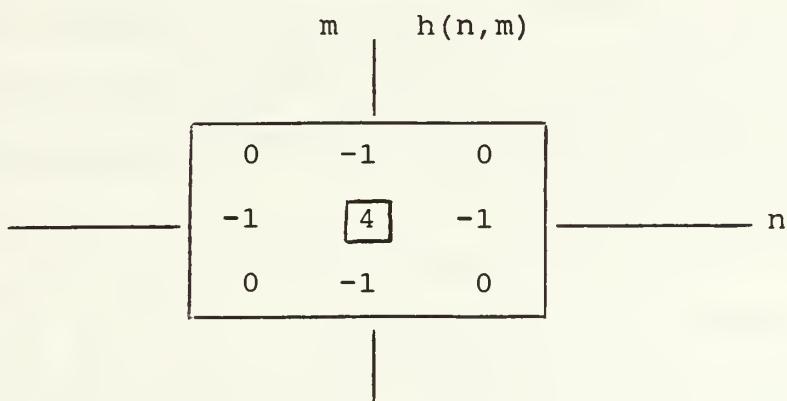


Figure 3-1 Laplacian Impulse Response

Its implementation involves convolving it with an image and is represented by the following difference equation:

$$y(n,m) = \sum_{i=-1}^1 \sum_{j=-1}^1 b_{ij} x(n-i, m-j) \quad (3.1)$$

where

$x(n,m)$  is the image input signal, and

$b_{ij}$  is the filter coefficient matrix  
( $b_{ij} = h(i,j)$ )

In the Z-transform domain this can be written as:

$$\begin{aligned} Y(z_1, z_2) &= H(z_1, z_2) \cdot X(z_1, z_2) \\ &= (4 - z^{-1} - z_1 - z^{-1} - z_2) \cdot X(z_1, z_2) \end{aligned}$$

In areas of an image where adjacent pixels have similar gray levels (low frequency, homogeneous areas), the result of this operator will be approximately zero. Where significant or sharp differences in gray levels between adjacent pixels exist, the result of this operation will be farther from zero. Thus the Laplacian difference operator is sometimes used as an "edge detector."

The problem addressed in this chapter is constructing the inverse of the operator. In the Z domain, the expression for the inverse would be [Ref. 4:p. 36]:

$$H^{-1}(z_1, z_2) = \frac{1}{H(z_1, z_2)} = \frac{1}{4 - z_1^{-1} - z_1 - z_2^{-1} - z_2}$$

which has an expansion as an infinite series of positive and negative powers of  $z_1$  and  $z_2$ . That is, considering this expression as a problem in long division, the result of such division would be an expression of the form:

$$H^{-1}(z_1, z_2) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} a_{ij} z_1^{-i} z_2^{-j}$$

where

$a_{ij}$  = coefficient values of  $z_1^i z_2^j$  resulting from the long division

Note that if

$$\frac{1}{4 - z_1^{-1} - z_1 - z_2^{-1} - z_2} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} a_{ij} z_1^{-i} z_2^{-j}$$

then cross multiplication would yield:

$$(4 - z^{-1} - z_1 - z^{-2} - z_2) \cdot \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} a_{ij} z^{-i} z^{-j} = 1 \quad (3.3)$$

The double summation expression in  $z_1$  and  $z_2$  will be truncated and considered to be an FIR filter with finite support and coefficients  $a_{ij}$ . This approximates the desired inverse filter. In particular, we will use the following constraints:

- 1) Choose the limits of summation to be equal in both directions, i.e.,

$$\sum_{i=-L}^L \sum_{j=-L}^L a_{ij} z_1^{-i} z_2^{-j}$$

This results in a "square" region of support for the filter (all values outside assumed zero).

- 2) Force the values for the filter coefficients to be symmetric, i.e.,  $a_{ij} = a_{-ij} = a_{i-j} = a_{-i-j} = a_{ji} = a_{-ji} = a_{j-i} = a_{-j-i}$ .

Using these constraints and implementing the cross multiplication equation (3.3) will result in an expression in  $z_1$  and  $z_2$ , with each combination of the  $z_1^i z_2^j$  terms having a coefficient whose form is a summation of terms in  $a_{ij}$

where the coefficients of  $a_{ij}$  are either 4 or -1. The coefficient of the  $z_1^0 z_2^0$  term must equal 1 and the coefficient of any other  $z_1^{-i} z_2^{-j}$  term must equal zero to satisfy equation (3.3).

As an example, let  $L = 1$ . Equation (3.3) can then be expressed as:

$$(4 - z^{-1} - z_1 - z^{-1} - z_2) \cdot \sum_{i=-1}^1 \sum_{j=-1}^1 a_{ij} z_1^{-i} z_2^{-j} \\ = 1 + 0 \cdot z_1^1 + 0 \cdot z_2^1 + 0 \cdot z_1^1 z_2^1 + \dots$$

Performing the double summation yields

$$\sum_{i=-1}^1 \sum_{j=-1}^1 a_{ij} z_1^{-i} z_2^{-j} = a_{-1-1} z_1^1 z_2^1 + a_{-10} z_1^1 z_2^0 + a_{-11} z_1^1 z_2^{-1} + \\ + a_{00} z_1^0 z_2^0 + a_{0-1} z_1^0 z_2^1 + a_{1-1} z_1^{-1} z_2^1 + a_{10} z_1^{-1} \\ + a_{01} z_2^{-1} + a_{11} z_1^{-1} z_2^{-1}$$

Performing the cross multiplication would yield 45 different terms in various combinations of  $z_1^{-i} z_2^{-j}$ . Combining these terms to find the coefficient expression for each  $z_1^{-i} z_2^{-j}$  term soon becomes rather tedious and impractical for even moderate values of  $L$ . An alternative way to proceed is to choose a  $z_1^a z_2^b$  term on the right hand side of the equation, and for each term in the expression  $4 - z_1^{-1} - z_1 - z_2^{-1} - z_2$ , determine what values of  $i$  and  $j$  are required so that when each term is multiplied by  $a_{ij} z_1^{-i} z_2^{-j}$ , it will result in an

expression in the chosen  $z_1^a z_2^b$  term on the right hand side of the equation. Choosing  $z_1^0 z_2^0$  ( $= 1$ ) on the right hand side of the equation, we have:

$$\begin{aligned} 4 \cdot a_{ij} z_1^{-i} z_2^{-j} &= cz_1^0 z_2^0 \text{ when } i = 0 \text{ and } j = 0; \text{ so } c = 4a_{00} \\ -z_1^{-1} \cdot a_{ij} z_1^{-i} z_2^{-j} &= cz_1^0 z_2^0 \text{ when } i = -1 \text{ and } j = 0; \text{ so } c = -a_{-10} \\ -z_1 \cdot a_{ij} z_1^{-i} z_2^{-j} &= cz_1^0 z_2^0 \text{ when } i = 1 \text{ and } j = 0; \text{ so } c = -a_{10} \\ -z_2^{-1} \cdot a_{ij} z_1^{-i} z_2^{-j} &= cz_1^0 z_2^0 \text{ when } i = 0 \text{ and } j = -1; \text{ so } c = -a_{0-1} \\ -z_2 \cdot a_{ij} z_1^{-i} z_2^{-j} &= cz_1^0 z_2^0 \text{ when } i = 0 \text{ and } j = 1; \text{ so } c = -a_{01} \end{aligned}$$

So the coefficient for  $z_1^0 z_2^0$  is simply a summation of the  $c$  terms obtained above, i.e.,

$$(4a_{00} - a_{-10} - a_{10} - a_{0-1} - a_{01}) z_1^0 z_2^0$$

This entire expression must equal 1 to satisfy (3.3), and since  $z_1^0 z_2^0 = 1$ ,  $4a_{00} - a_{-10} - a_{10} - a_{0-1} - a_{01} = 1$  also.

Using the same method for the  $z_1^1 z_2^1$  term on the right hand side yields:

$$\begin{aligned} 4 \cdot a_{ij} z_1^{-i} z_2^{-j} &= cz_1^1 z_2^1 \text{ when } i = -1 \text{ and } j = -1; \text{ so } c = 4a_{-1-1} \\ -z_1^{-1} \cdot a_{ij} z_1^{-i} z_2^{-j} &= cz_1^1 z_2^1 \text{ when } i = -2 \text{ and } j = -1; \text{ so } c = -a_{-2-1} \\ -z_1 \cdot a_{ij} z_1^{-i} z_2^{-j} &= cz_1^1 z_2^1 \text{ when } i = 0 \text{ and } j = -1; \text{ so } c = -a_{0-1} \\ -z_2^{-1} \cdot a_{ij} z_1^{-i} z_2^{-j} &= cz_1^1 z_2^1 \text{ when } i = -1 \text{ and } j = -2; \text{ so } c = -a_{-1-2} \\ -z_2 \cdot a_{ij} z_1^{-i} z_2^{-j} &= cz_1^1 z_2^1 \text{ when } i = -1 \text{ and } j = 0; \text{ so } c = -a_{-10} \end{aligned}$$

The resulting term in  $z_1^1 z_2^1$  is:

$$(4a_{-1-1} - a_{-2-1} - a_{0-1} - a_{-1-2} - a_{-10}) z_1^1 z_2^1$$

This expression must equal zero, since there is no  $z_1^1 z_2^1$  term on the right side of (3.3), so  $4a_{-1-1} - a_{-2-1} - a_{0-1} - a_{-1-2} - a_{-10} = 0$ .

This procedure can be extended to any number of  $z_1^{-i} z_2^{-j}$  terms. When this is done, the resulting expressions for the coefficients of  $z_1^{-i} z_2^{-j}$  can be formed into a set of simultaneous equations in order to solve for the  $a_{ij}$  coefficient values. However, due to the symmetry condition imposed above, some of the equations for the coefficients of the  $z_1^{-i} z_2^{-j}$  terms are linearly dependent. For values of  $i$  and  $j$  that yield unique or distinct values for  $a_{ij}$ , the resulting  $z_1^{-i} z_2^{-j}$  coefficient expressions are linearly independent. For example, the coefficient expression for the  $z_1^{-1} z_2^0$  term is linearly independent of the coefficient expression for the  $z_1^{-1} z_2^{-Q}$  term, since  $a_{10} \neq a_{11}$ . But the coefficient expression for the  $z_1^1 z_2^0$  term is linearly dependent on the coefficient expression for the  $z_1^0 z_2^{-1}$  term, since  $a_{10} = a_{01}$ . Using only the linearly independent equations for a given filter size yields a set of  $p$  equations in  $p$  unknowns, where  $p$  is the number of unique and distinct  $a_{ij}$  values in the inverse filter. The value of  $p$  is related to the size of the desired inverse filter. If the size of the filter is  $N \times N$ ,

the number of unique  $a_{ij}$  values using the symmetry of constraint above is:

$$p = (\frac{N-1}{2} + 1) + (\frac{N-1}{2}) + (\frac{N-1}{2} - 1) + \dots + 1 \quad (3.4a)$$

$$= \frac{(N+1)(N+3)}{4} \quad (3.4b)$$

For example, with  $N = 7$ , the unique  $a_{ij}$  values in a  $7 \times 7$  inverse filter can be represented by  $a_{33}, a_{32}, a_{31}, a_{30}, a_{22}, a_{21}, a_{20}, a_{11}, a_{10}, a_{00}$ . Though there are 49 elements in a  $7 \times 7$  filter, all of them are equal to one of these values listed, due to symmetry. Obviously,  $N$  is constrained to be odd for a square filter with a unique element  $a_{00}$  in the center.

The solution of the resulting  $p$  equations yields the values for the  $p$  filter elements or coefficients. This defines the FIR approximation to the inverse filter. It is only an approximation due to the finite size constraint imposed, and it might be expected that the larger the filter size, the better the approximation.

An algorithmic procedure for obtaining the  $a_{ij}$  coefficients is outlined below. An example follows.

- 1) Determine the desired size of the inverse filter.
- 2) For each combination of (positive)  $i, j$  values corresponding to a unique  $a_{ij}$  filter coefficient, identify the five term summation equation associated with each  $z_1^{-1} z_2^{-j}$  term.

- 3) Combine equal  $a_{ij}$  values and develop a matrix of coefficients for the  $a_{ij}$  values. Let this matrix be  $A$ .
- 4) Denoting the vector of unique  $a_{ij}$  values as  $\bar{a}$ , the set of simultaneous equations in matrix form is:

$$\bar{A}\bar{a}^T = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \quad (3.5)$$

where

$$\bar{a} = [a_{00} \ a_{10} \ a_{11} \ a_{20} \ a_{21} \ a_{22} \dots a_{\frac{N-1}{2} \frac{N-1}{2}}]$$

The top row of  $\bar{A}$  corresponds to the summation of  $a_{ij}$  terms that represents the coefficient of the  $z_1^0 z_2^0$  term.

- 5) Solve (3.5) for  $\bar{a}$ .

An example of this procedure is appropriate at this point. For an inverse filter of size  $N \times N$ :

Step 1

Let  $N = 5$  (therefore  $L = 2$ )

Step 2

The coefficients corresponding to each unique  $z^{-i} z^{-j}$  term are:

$$\begin{aligned} z_1^0 z_2^0 &\rightarrow 4a_{00} - a_{10} - a_{-10} - a_{01} - a_{0-1} \\ z_1^{-1} z_2^0 &\rightarrow 4a_{10} - a_{20} - a_{00} - a_{11} - a_{1-1} \\ z_1^{-1} z_2^{-1} &\rightarrow 4a_{11} - a_{21} - a_{01} - a_{12} - a_{10} \end{aligned}$$

$$\begin{aligned}
 z_1^{-2}z_2^0 &\rightarrow 4a_{20} - a_{30} - a_{10} - a_{21} - a_{2-1} \\
 z_1^{-2}z_2^{-1} &\rightarrow 4a_{21} - a_{31} - a_{11} - a_{20} - a_{2-2} \\
 z_1^{-2}z_2^{-2} &\rightarrow 4a_{22} - a_{32} - a_{12} - a_{23} - a_{21}
 \end{aligned}$$

### Step 3

Combining equal terms in Step 2 and expressing the coefficients of  $a_{ij}$  in matrix form yields an  $\bar{A}$  matrix of:

$$\bar{A} = \begin{bmatrix} 4 & -4 & 0 & 0 & 0 & 0 \\ -1 & 4 & -2 & -1 & 0 & 0 \\ 0 & -2 & 4 & 0 & -2 & 0 \\ 0 & -1 & 0 & 4 & -2 & 0 \\ 0 & 0 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -2 & 4 \end{bmatrix}$$

### Step 4

With  $\bar{A}$  given in Step 3, the  $\bar{a}$  vector for (3.5) is

$$\bar{a} = [a_{00} \ a_{10} \ a_{11} \ a_{20} \ a_{21} \ a_{22}]$$

### Step 5

Solving (3.5) for  $\bar{a}$  involves inverting  $\bar{A}$  and multiplying it by  $[1 \ 0 \ 0 \ 0 \ 0 \ 0]^T$ . Thus:

$$\bar{a} = \bar{A}^{-1} \cdot [1 \ 0 \ 0 \ 0 \ 0 \ 0]^T$$

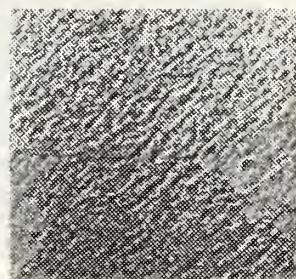
Appendix H illustrates the forms of the resulting inverse filters of various sizes, as well as the normalized and unnormalized filter cross sections.

One way to validate the resulting inverse filter is to convolve it with the original Laplacian difference operator. The result should approximate an impulse at the origin. Appendix I shows the results of this convolution using  $3 \times 3$ ,  $5 \times 5$ ,  $7 \times 7$ ,  $9 \times 9$ ,  $15 \times 15$ , and  $21 \times 21$  size inverse filters. It is seen there that as the size of the filter gets larger, it becomes a better approximation to the true inverse and the convolution looks more like an impulse.

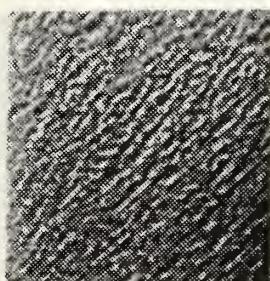
To test the application of this filter on an actual image, a test image was filtered using the Laplacian difference operator. Then the resulting image data were filtered again using various size inverse filters. The results are shown in Figures 3-2 and 3-3. Note that the image resulting from Laplacian FIR filtering seems more stationary than the test image, which was one of the desired results. Inverse filtering of that result yields images that are progressively more similar to the original test image as the size of the inverse filter increases. However, a rather large inverse filter is needed to accurately reproduce the image. The result of the  $21 \times 21$  inverse filter is quite similar to the test image, with some loss of contrast or darkness in certain areas, but with essentially the same pattern. The effect of the size limitation of the



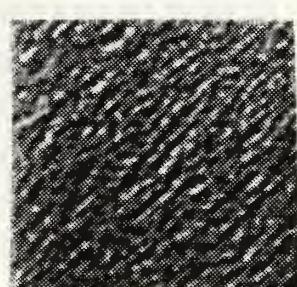
Test image



Laplacian filtered test image

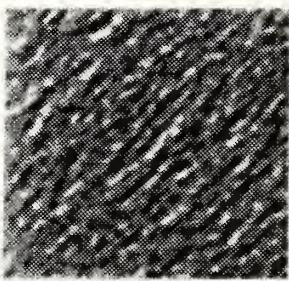


$3 \times 3$  inverse filter

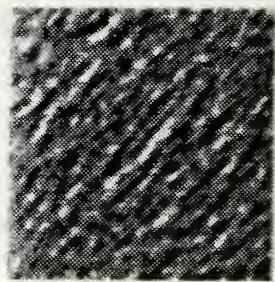


$5 \times 5$  inverse filter

Figure 3-2 Results of Filtering Test Image with Laplacian FIR Filter and its Inverse



$7 \times 7$  inverse filter



$9 \times 9$  inverse filter



$15 \times 15$  inverse filter



$21 \times 21$  inverse filter

Figure 3-3 Results of Filtering Test Image with Laplacian FIR Filter and its Inverse

inverse filter, as manifested in the convolution of the Laplacian and its inverse above, would seem to explain the lack of perfect test image reproduction. Larger inverse filter sizes could be tried, but large inverse filter sizes relative to image size would result in a significant portion around the edge of the image having only a part of the filter applied to it. This would adversely affect the overall quality of image reproduction.

#### IV. APPLICATION OF THE ARIMA MODEL TO IMAGE TEXTURES

As outlined in Chapter I, the utility of the ARIMA model centers around the fact that a difference operator applied to an image texture may improve the stationarity of the image statistical characteristics. A stationary image texture is required for accurate modeling by autoregressive techniques, and it was hoped that application of the autoregressively generated texture to an approximate inverse of the difference operator may yield a more accurate or recognizable representation of the original nonstationary image, as compared to a purely autoregressively generated version.

##### A. APPLICATION OF LAPLACIAN INVERSE FILTER TO AUTO-REGRESSIVELY GENERATED IMAGES

As an initial examination of the effects of the inverse filter developed in Chapter III on image textures, selected images generated in Chapter II were input to the  $21 \times 21$  version of that filter. Figures 4-1 through 4-3 illustrate the results. All attempts resulted in a blurred or smoothed version of the original image. Since the inverse of a difference operation is a summation or "integration" operation, and since integration operations can be expected to smooth or blur (low pass filter) signals [Ref. 8:pp. 136-154], the results are not surprising. However,

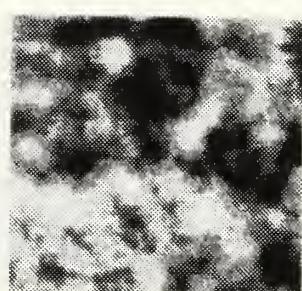


Figure 4-1 Images from Figure 2-7 (top 4) Applied to Summation Filter

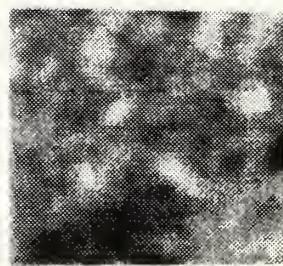
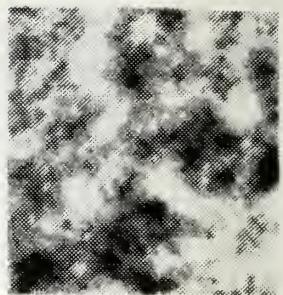
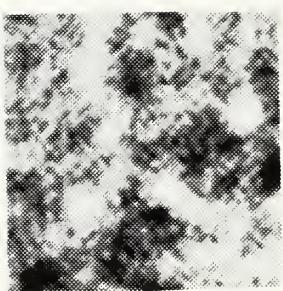


Figure 4-2 Images from Figure 2-11 (top 4) Applied to Summation Filter

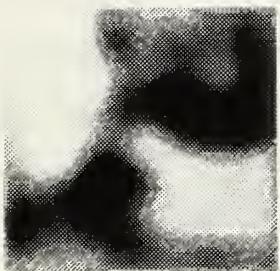


Figure 4-3 Images from Figure 2-14 (top 4) Applied to Summation Filter

except to the extent that blurring is useful or desirable, applying the summation filter to image signals that are not based on the application of the corresponding difference filter to that signal seems to be of little utility.

In the remainder of this chapter we consider application of the summation filter to regenerate actual image textures.

#### B. AUTOREGRESSIVE FILTER PARAMETER ESTIMATION PROCEDURES

The first step in testing the ARIMA model is to estimate the autoregressive, quarter plane filter parameters required to model the real image textures and the signal resulting from application of the Laplacian operator to those images. For a zero-mean signal, these model parameters are found by solving a set of Normal equations. In these equations the white noise covariance is referred to as the prediction error covariance. The Normal equations can be expressed as

$$\bar{R} \begin{bmatrix} \bar{a}_0 \\ \bar{a}_1 \\ \vdots \\ \vdots \\ \bar{a}_{p-1} \end{bmatrix} = \begin{bmatrix} \bar{s} \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \quad (4.1)$$

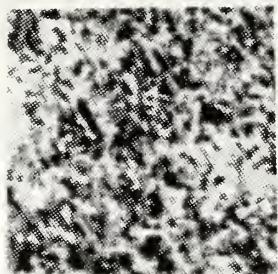
where the  $\bar{R}$  matrix is the correlation matrix for the signal (block Toeplitz with Toeplitz blocks), the  $\bar{a}$  vector consists of appropriately ordered filter coefficient vectors, and  $\bar{s}$  is a vector containing the prediction error covariance as

the first and only nonzero element. Here  $\bar{a}_i = [a_{i0} \ a_{i1} \ a_{i2} \ \dots \ a_{i, Q-1}]^T$  and  $\bar{s} = [\sigma^2 \ 0 \ 0 \ \dots \ 0]^T$ .

Calculating the correlation matrix and prediction error covariance from the image signals, and solving (4.1) for the  $\bar{a}$  vectors, provides all the parameters needed for the 2-D AR model. The multichannel form of the Levinson recursion can be used to solve these equations more efficiently [Ref. 2:p. 454].

### C. APPLICATION TO REAL IMAGE TEXTURES

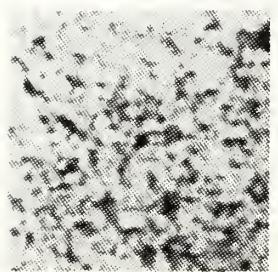
Actual image textures used here are from the image data base at the University of Southern California's Signal and Image Processing Institute [Ref. 9:pp. 13-14]. The images selected from this data base are contained in a book by Brodatz [Ref. 10]. Portions of the images of size  $128 \times 128$  pixels were obtained and used as a basis for processing. Filter coefficients were calculated for the real image textures shown in Figures 4-4 and 4-5. Various filter sizes were tried to determine which yielded the best results in generating a particular image, and a quarter-plane filter size of  $4 \times 4$  was selected. Results of autoregressive filtering of white noise using the appropriate calculated coefficients to model each texture are given in Figures 4-6 and 4-7. Images generated by applying the Laplacian difference operator to the real images are shown in Figures 4-8 and 4-9. Autoregressive generation of these images using



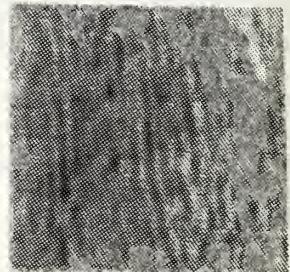
Grass



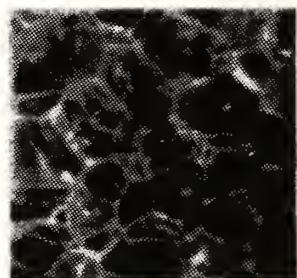
Bark



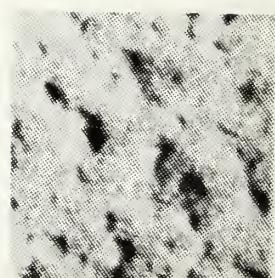
Sand



Water

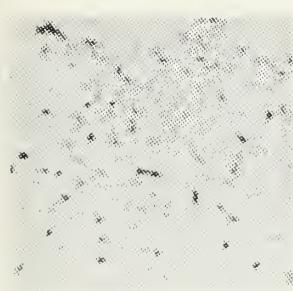


Bubbles



Wall

Figure 4-4 Actual Image Textures



Sand

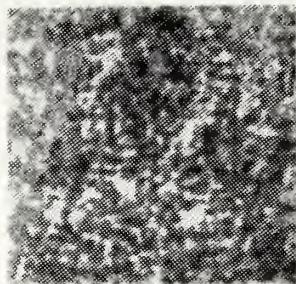


Gravel

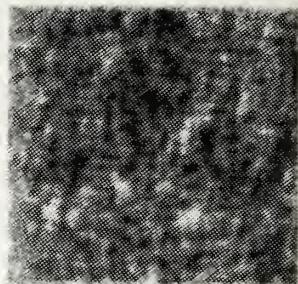


Grass

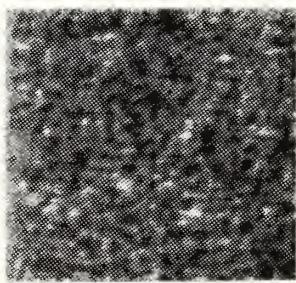
Figure 4-5 Actual Image Textures



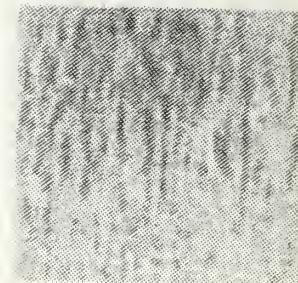
Grass



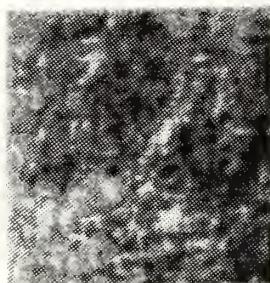
Bark



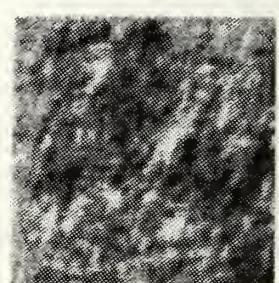
Sand



Water

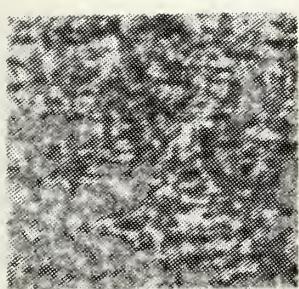


Bubbles



Wall

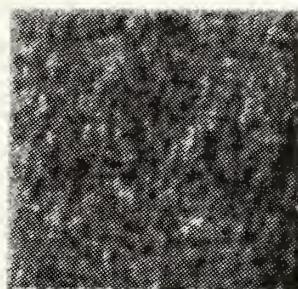
Figure 4 -6 Image Textures Generated Using an AR Model



Sand

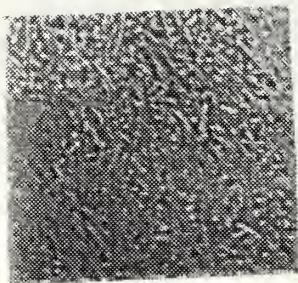


Gravel

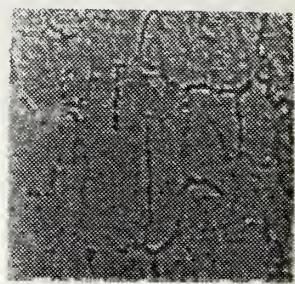


Grass

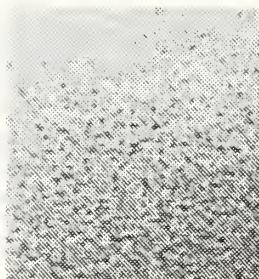
Figure 4-7 Image Textures Generated Using an AR Model



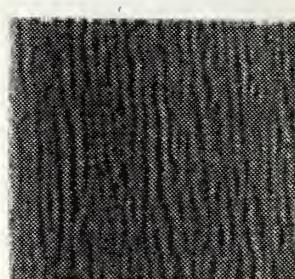
Grass



Bark



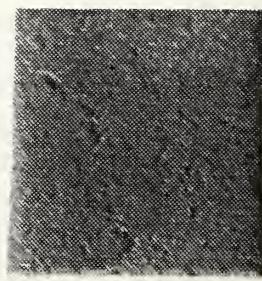
Sand



Water

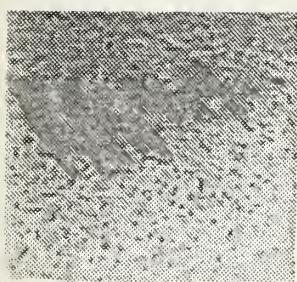


Bubbles

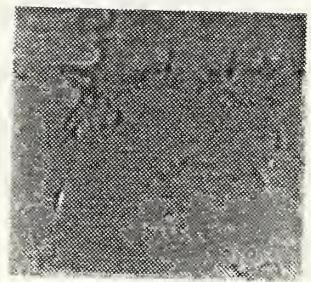


Wall

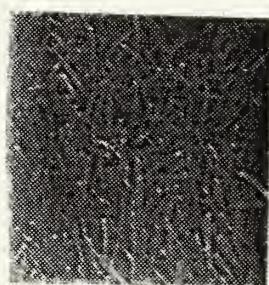
Figure 4-8 Actual Images After Laplacian Filtering



Sand



Gravel



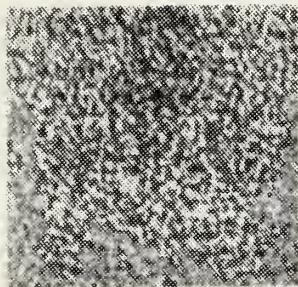
Grass

Figure 4-9 Actual Images After Laplacian Filtering

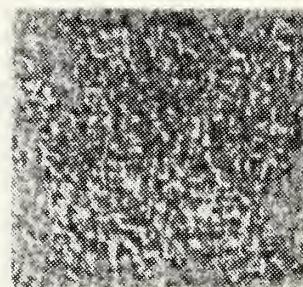
filters with the corresponding calculated coefficients are given in Figures 4-10 and 4-11. Finally, the application of the signal represented by the images in Figures 4-10 and 4-11 (without the 0-255 scaling reflected in these figures) to the  $21 \times 21$  inverse filter described in Chapter III yields the images shown in Figures 4-12 and 4-13. Comparison of all of the above results yields the following observations:

- 1) Autoregressive modeling of the water, grass and sand textures yielded good results. Some of the other textures with more structure and sharp local variations were not reproduced well.
- 2) Autoregressive reproduction of images created after application of the difference operator, with the exception of the water image, yielded generally poor results. As observed in Chapter III, the application of the difference operator produces a seemingly more stationary result, but the edge structure that remained in most of the images after application of the difference operator was in general not reproducible using a purely AR model.
- 3) Application of the inverse filter to the image signal generated by AR model reproduction of the difference operator results yielded smoothed versions of those results. This is similar to what was observed in Section A of this chapter when images were applied to the inverse filter that were not based on the specific data generated by the difference operator.

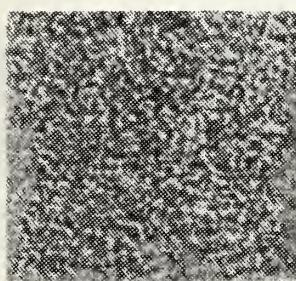
As a final test of the ARIMA model, a  $64 \times 64$  contrast enhanced aerial photograph of trees, with smoother variations and in general less edge structure than the other images tested, was tried. The results are shown in Figure 4-14. Though this image seemed somewhat better adapted to the model, overall the same observations outlined above apply.



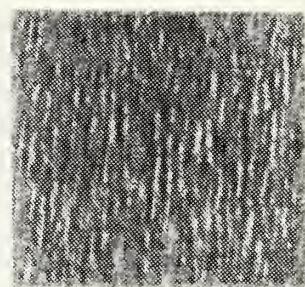
Grass



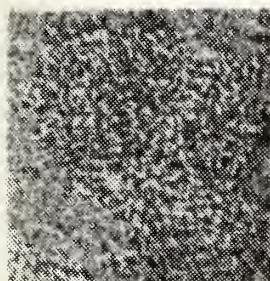
Bark



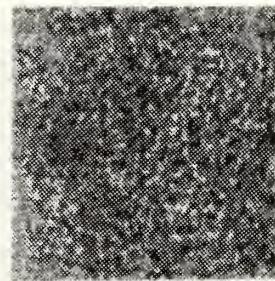
Sand



Water

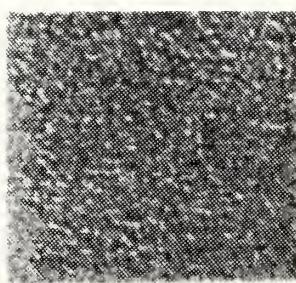


Bubbles

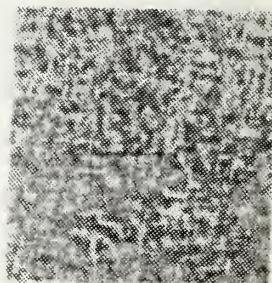


Wall

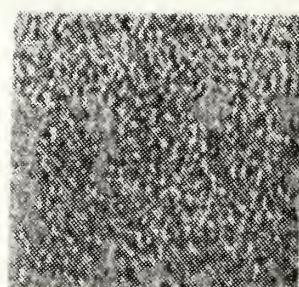
Figure 4-10 Laplacian Filtered Image Textures Generated Using an AR Model



Sand

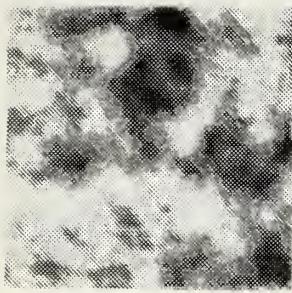


Gravel

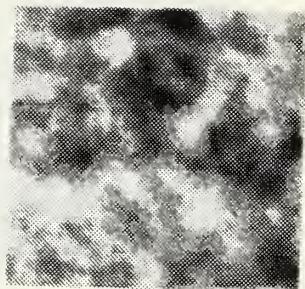


Grass

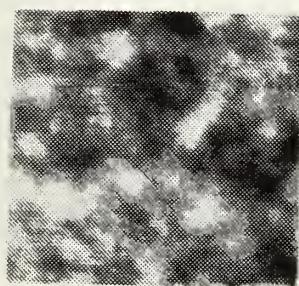
Figure 4-11 Laplacian Filtered Image Textures Generated Using an AR Model



Grass



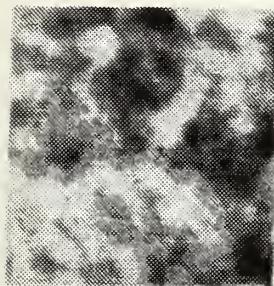
Bark



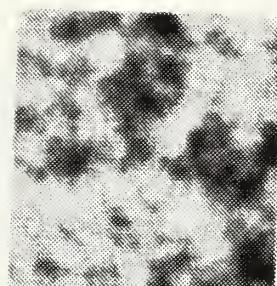
Sand



Water



Bubbles



Wall

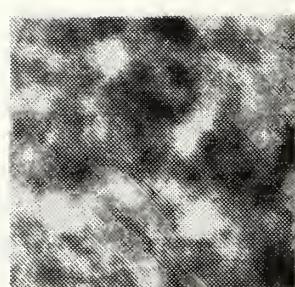
Figure 4-12 Image Textures Generated Using an ARIMA Model



Sand

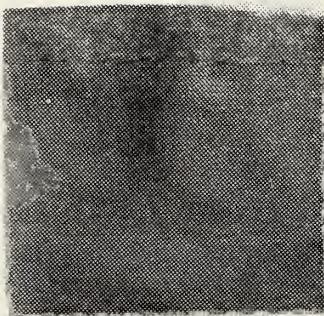


Gravel



Grass

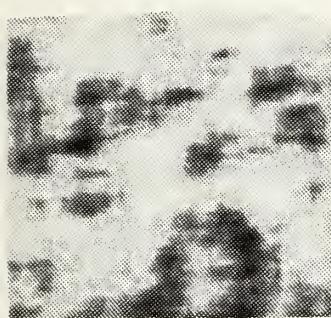
Figure 4-13 Image Textures Generated Using an ARIMA Model



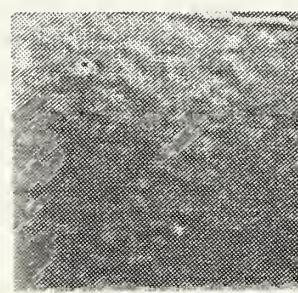
Original Image



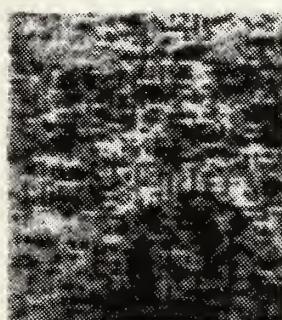
Contrast Enhanced



AR Generated  
Contrast Enhanced  
Image



Difference Operator  
Result



AR Generated  
Difference Operator  
Result



ARIMA Result

Figure 4-14 Final Test of ARIMA Model on Contrast Enhanced Trees (Magnification X2)

#### D. SUMMARY

The effectiveness of AR reproduction of image data using white noise input and filter coefficients calculated based on the statistics of the image signal is highly dependent on the nature of the image data. The water image, for example, with its smoothly varying and rather homogeneous nature, was quite well adapted to AR reproduction. Other images with more structure, abrupt variations, and more non-homogeneous characteristics, were not autoregressively reproducible to any great extent.

Using the ARIMA model, it seems that the operation of the inverse filter is very sensitive to the nature of the input data. Input data that are strictly based on the difference operator output can reproduce the original image, as was found in Chapter III. However, the AR model used to generate the inverse filter input (based on the statistics of image signal produced using the difference operator) does not generate image data accurately enough to reproduce images that resemble the real images tested.

## V. CONCLUSIONS

This thesis sought to explore experimentally and to understand how linear filtering models could be used to generate texture in images. Of particular interest was the investigation of 2-D ARIMA models to see if they might be of any utility in this effort. Some time was spent exploring separable 2-D models to understand how transfer function pole placement affected image texture characteristics. Image textures generated using these models and applied to the summation filter yielded blurred or smooth textures with seemingly little variety or utility. The ultimate test of the model was its ability to reproduce actual image textures. The purely AR portion of the model reproduced a few types of actual textures well. However, the full ARIMA model failed to generate image textures that resembled the source images used. Many of the textures had strong edge differences that were not accurately reproducible by the AR model. Also, the summation filter developed seemed very sensitive to deviations in image signal data from that generated by application of the difference operator; that is, the procedure seemed not to be "robust."

Since many of the images tested here have definite edge structure, the difference image had lines which were not reproduced well by the AR model. Correspondingly the

integrated AR model did not reproduce the original image. For this type of image, a combination of a line point process model [Ref. 11] with the integrator, would possibly have been more suitable. The image of trees had not such edge structure and produced somewhat better results. Further experimentation with images of this type and the ARIMA model would perhaps be appropriate.

## APPENDIX A

### COMPUTER PROGRAMS, SUBROUTINES, AND FUNCTIONS

Listed below are the names, associated computer systems, and functions of the various computer algorithms used to accomplish this thesis research. All programs were written by the thesis author unless otherwise noted. Program source codes are given at the end of this appendix (except for the MAKFIL\* series).

#### A. PROGRAMS

##### 1. AUTOREG (VAX/VMS FORTRAN)

The program did the following:

- 1) Generated a  $128 \times 128$  zero mean white noise matrix using subroutine PGAUSS.
- 2) Multiplied the white noise by the appropriate image data standard deviation when necessary.
- 3) Converted that matrix into a displayable image file using subroutines SCALE and INTBYTE, when necessary.
- 4) Read filter parameters from an input file into an array.
- 5) Implemented the equation:

$$y(n,m) = - \sum_{\substack{i=0 \\ (i,j) \neq 0}}^{P-1} \sum_{j=0}^{Q-1} a_{ij} y(n-i, m-j) + w(n, m) \quad (A.1)$$

using the white noise array and the filter coefficient array as inputs.

- 6) Converted the array result from 4) into a displayable image file using subroutines SCALE and INTBYTE.
- 7) Used subroutine SUBINTFILE to create image data files from filter results for further processing.
- 8) Used subroutine NONC to apply the summation filter to image data when necessary.

2. NONCAUSAL (VAX/VMS FORTRAN)

This program did the following:

- 1) Read filter coefficient values into an array.
- 2) Read image data from an input image file, converted it to integer values using subroutine BYTEINT, calculated the mean from the data, and placed the data into a real array.
- 3) Implemented the equation:

$$y(n,m) = \sum_{i=-L}^L \sum_{j=-L}^L a_{ij} x(n-i, m-j) \quad (A.2)$$

using the image data array and the filter coefficient array.

- 4) Called the subroutine NONC to implement the equation in Step 3) a second time, when necessary.
- 5) Converted the result of Step 3) to a displayable image file using subroutine SCALE and INTBYTE.

3. CONV (VAX/VMX FORTRAN)

This program performs the same basic functions as NONCAUSAL, without having the capability of calling subroutine NONC. It was used for convenience in convolving certain filter structures with certain test images directly.

4. MAKFIL\* (VAX/VMS FORTRAN)

This family of programs was used to create various autoregressive and FIR filter coefficient files, using source data manually entered into the program.

5. SPECOR2 (IBM SYSTEM/370 3033 VS FORTRAN 1.4.1)

This program implemented the equations derived in Appendix B and created data files used in developing the corresponding graphs.

6. SPECOR3 (IBM SYSTEM/370 3033 VS FORTRAN 1.4.1)

This program implemented the equations derived in Appendix D and created data files used in developing the corresponding graphs.

7. SPECOR3A (IBM SYSTEM/370 3033 VS FORTRAN 1.4.1)

This program implemented the equations derived in Appendix F and created data files used in developing the corresponding graphs.

8. VARIMGS (VAX/VMS FORTRAN)

This program was used to display image data files on the COMTAL (not written by author).

9. PIECE (VAX/VMS FORTRAN)

This program was used to make  $128 \times 128$  image data files from larger image data files.

10. INTFILE (VAX/VMS FORTRAN)

This program created appropriately formatted integer files from input image data for further processing.

## 11. TRANS (VAX/VMS FORTRAN)

This program changed the format of filter coefficient data files into a form readable by the image processing programs.

## 12. NSHP (VAX/VMS FORTRAN)

This program was used to convert quarter-plane autoregressive filter coefficient data to non-symmetric half-plane autoregressive filter coefficient data based on the transformation outlined in Chapter II, Section D.

## B. SUBROUTINES

### 1. PGAUSS (VAX/VMS FORTRAN)

This subroutine, written by C.W. Therrien, was used to generate zero mean, unit variance white noise using RAN (a random number generator function) SQRT, COS, and SIN FORTRAN functions.

### 2. SCALE (VAX/VMS FORTRAN)

This subroutine takes an image data array and converts it to an integer array with values between an input maximum (MAX) and minimum (MIN) using the following scaling formula:

$$I(i,j) = \frac{(A(i,j)-LOW) \times (MAX-MIN)}{HIGH-LOW} + MIN \quad (A.3)$$

$A(i,j)$  is the input image data array,  $I(i,j)$  is the output integer array, and HIGH and LOW are the high and low values of  $A(i,j)$ , respectively (calculated in this subroutine).

This is done to provide appropriate values for image files that will be displayed on the COMTAL Vision One/20, since the gray scale intensity level of each pixel is represented by an 8-bit word. So values possible (in base 10) range from 0 (darkest), to 255 (brightest).

### 3. INTBYTE AND BYTEINT (VAX/VMS FORTRAN)

These subroutines are necessary since data in an image file are stored in two's complement form. The related variable type in FORTRAN for these values is BYTE. To process image data using FORTRAN implementation of the appropriate formulas, these byte values must be converted to integer (and eventually real using the FLOAT function) form. Results of image processing formulas in real form must be converted to integer (using the INT function) and then byte form to be placed in image data files. INTBYTE converts integer type variables to byte type variables using the following criterion (I is an integer and B is a byte):

If  $I \leq 127$  and  $I \geq 0$  then  $B = I$

If  $I > 127$  and  $I \leq 255$  then  $B = I - 256$

BYTEINT converts byte type variables to integer type variables using the following criterion:

If  $B \geq -128$  and  $B < 0$  then  $I = B + 256$

If  $B \geq 0$  and  $B \leq 127$  then  $I = B$

### 4. SUBINTFILE (VAX/VMS FORTRAN)

This subroutine performed the same function as INTFILE, but could be called by a program to operate on

processed image data arrays, rather than just image file data inputs.

#### 5. NONC (VAX/VMX FORTRAN)

This subroutine performs essentially the same functions as NONCAUSAL, except that it can be called by a program to operate on an image data array.

### C. APL FUNCTIONS

The APL systems on the IBM System/370 3033 and VAX/UNIX were used for matrix manipulations and operations, for graphing filter structures and convolution results, and for calculating autoregressive filter coefficients from image data. All APL functions except MAKMAT were written by C.W. Therrien.

#### 1. MAKMAT (IBM)

This function was used to create the large coefficient matrices ( $\bar{A}$ ) used in calculating the FIR filter coefficients as outlined in Chapter III.

#### 2. CC2 (IBM)

This function was used to circularly convolve the Laplacian FIR filter and its various inverses. Appropriate zero-padding of these filters makes the resulting circular convolution equivalent to linear convolution [Ref. 2:pp. 70-72], which was the desired operation.

#### 3. GETDATA (VAX/UNIX)

This function is used to transfer image data files from a UNIX subdirectory to an APL workspace.

4. PUTDATA (VAX/UNIX)

This function is used to transfer filter coefficient data files from an APL workspace to a UNIX subdirectory.

5. MEAN (VAX/UNIX)

This function is used to calculate the mean of an image data file for use in the APL function COVF.

6. COVF (VAX/UNIX)

This function is used to calculate terms in the 2-D covariance function for use in the APL function CORR.

7. CORR (VAX/UNIX)

This function is used to estimate the 2-D covariance function of the image data.

8. MVLEV (VAX/UNIX)

This function is used in APL function FF2DLEV to calculate necessary parameters for the 2-D Levinson recursion from the covariance function of the image data.

9. FF2DLEV (VAX/UNIX)

This function performs the 2-D Levinson recursion to solve for the filter coefficient vector.

## AUTOREG

```

c THIS PROGRAM GENERATES AN IMAGE TEXTURE USING WHITE NOISE AS AN INPUT
c TO AN AUTOREGRESSIVE FILTER WHOSE PARAMETERS ARE OBTAINED FROM THE
c FILE FILCOEF. SUBROUTINE PGAUSS IS USED TO GENERATE THE INPUT WHITE
c NOISE AND SUBROUTINES SCALE AND INTBYTE ARE USED TO PREPARE IMAGE
c DATA ARRAYS FOR DISPLAY. SUBROUTINE SURINVFILE IS USED IF AN
c INTEGER FILE RESULT IS DESIRED, AND SUBROUTINE VONC IS USED IF A
c LAPLACIAN INVERSE FILTERING STEP IS NEEDED.
c
c DEFINE VARIABLES
    byte a(0:127),bim(0:127,0:127)
    integer n,seed,rsize,csize,om1,om1,i,j,jol1,row,col,integ(0:127,0:1
    :27)
    real*8 val1,val2,wn(0:127,0:127),dsum,in(0:127,0:127),coef(0:9,0:9
    ),var,max,min,inout(0:127,0:127)
c OPEN FILES
    open(unit=1,name='frathmann.dat',an,dat',type='new',access='direct
    ',recordsize=32,maxrec=129)
    open(unit=2,n3me='frathmann.im1final9a.dat',type='new',access='di
    rect',recordsize=32,maxrec=129)
    open(3,file='frathmann.tatal fcdoub.dat',status='old')
c DEFINE PARAMETERS
    seed=1234567
    rsize=127
    csize=127
    om1=3
    om1=3
    max=229.0d0
    min=27.0d0
c CREATE WHITE NOISE ARRAY
    do 10 i=0,rsize
        do 20 j=0,csize-1
            jol1=i+1
            call pgauss(seed,val1,val2)
            wn(i,j)=val1
            wn(i,jol1)=val2
20    continue
10    continue
c SCALE ARRAY WN AND CONVERT TO BYTE FORM
    call scale(wn,integ,max,min)
    call intbyte(integ,bim)
c WRITE THE WHITE NOISE IMAGE ARRAY TO A FILE
    do 30 i=0,rsize
        do 40 j=0,csize
            a(i,j)=bim(i,j)
40    continue
    write(1'i+1) (a(n),n=0,rsize)
30    continue
c READ FILTER PARAMETERS INTO AN ARRAY
    do 50 i=0,om1
        do 60 j=0,om1
            read(3,551) coef(i,j)
60    format(120.12)
50    continue
55    continue
c VU_TTPLY WHITE NOISE BY REAL IMAGE STANDARD DEVIATION
    var=sqrt(coef(0,0))
    do 70 i=0,rsize
        do 80 j=0,csize
            wn(i,j)=wn(i,j)*var
80    continue

```

## AUTOREG (CONT.)

```

70 continue
c APPLY WHITE NOISE TO THE AUTOREGRESSIVE FILTER
    do 110 n=0,rsize
        do 120 m=0,csize
            asum=0.0d0
            do 130 i=0,nm1
                do 140 j=0,cm1
                    if((i.eq.0).and.(j.eq.0)) do to 140
                    rn=n-1
                    col=rn-j
                    if((row.lt.0).or.(col.lt.0)) do to 140
                    asum=(coeff(i,j)*im(row,col))+asum
140        continue
130        continue
    i=(n,m)=(-1.0d0*asum)+wn(n,m)
120        continue
110        continue
c FILTER THE IMAGE DATA ARRAY USING THE LAPLACIAN INVERSE FILTER
    call nonc(in,out)
c SCALE THE RESULTING IMAGE ARRAY AND CONVERT TO BYTE FORM
    call scale(in,out,max,min)
    call intbyte(out,min)
c WRITE THE GENERATED IMAGE INTO A FILE
    do 150 i=0,rsize
        do 160 j=0,csize
            a(j)=bim(i,j)
160        continue
            write(2'i+1) (a(n),n=0,rsize)
150        continue
c CLOSE FILES
    close(unit=1)
    close(unit=2)
    close(3)
    stop
end

```

## NONCAUSAL

```

c THIS PROGRAM GENERATES AN IMAGE TEXTURE USING A NONCAUSAL FIR FILTER
c THESE PARAMETERS ARE OBTAINED FROM A DATA FILE. THE FILTER IS
c APPLIED TO AN IMAGE. SUBROUTINES SCALE, INTBYTE, AND BYTEINT ARE
c USED TO PREPARE IMAGE DATA ARRAYS FOR DISPLAY. SUBROUTINE NONC
c IS USED IF AN INTERMEDIATE FILTERING STEP IS DESIRED.
c
c DEFINE VARIABLES
    byte a(0:127),bim(0:127,0:127)
    integer n,rsize,csize,i,j,row,col,inteo(0:127,0:127),fsize,nsrsize,
    index1,index2
    real*8 rsum,in(0:127,0:127),coeff(-10:10,-10:10),nsum,mean,dum(0:1
    27,0:127),high,low,try(-1:128,-1:128),max,min
c OPEN FILES
    open(unit=1,name='frathmann.inldark.dat',type='old',access='dir
    ect',recordsize=32,maxrec=128)
c     open(unit=2,name='frathmann.imlarinb.dat',type='new',access='dire
    ct',recordsize=32,maxrec=128)
    open(3,file='frathmann.dat\ncfilco3a.dat',status='old')

```

## NONCAUSAL (CONT.)

```

c  DEFINE PARAMETERS
    index1=-1
    index2=128
    rsize=127
    csize=127
    fsize=1
    nsize=-1*fsize
    max=255.0d0
    min=0.0d0
c  READ FILTER PARAMETERS INTO AN ARRAY
    do 10 i=nsize,fsize
        do 20 j=nsize,fsize
            read(3,25) coef(i,j)
25        format(d20.12)
20        continue
10    continue
c  READ IMAGE TO BE FILTERED INTO AN ARRAY AND CONVERT TO INTEGER
    do 30 i=0,rsize
        read(1,i+1) (a(n),n=0,127)
        do 40 j=0,csize
            bin(i,j)=a(j)
40        continue
30    continue
    call systint(bin,integ)
c  CONVERT THE INTEGER ARRAY INTO A REAL ARRAY AND COMPUTE THE MEAN
    nsun=0.0d0
    high=17000.0d0
    low=10000.0d0
    do 50 i=0,rsize
        do 50 j=0,csize
            im(i,j)=float(integ(i,j))
            nsun=im(i,j)+nsun
            if(im(i,j).gt.high) high=im(i,j)
            if(im(i,j).lt.low) low=im(i,j)
50    continue
50    continue
    mean=nsun/(128.0*128.0)
    write(*,55) mean,high,low
55    format(' original image data - mean=',d12.5,' high=',d12.5,' low=',d
           12.5)
c  SUBTRACT THE MEAN FROM THE DATA
    do 70 i=0,127
        do 80 j=0,127
            in(i,j)=im(i,j)-mean
80        continue
70    continue
c  FILTER THE IMAGE ARRAY
    do 90 n=index1,index2
        do 100 m=index1,index2
            dsum=0.0d0
            do 110 i=nsize,fsize
                do 120 j=nsize,fsize
                    row=n-i
                    col=m-j
                    if((row.lt.0).or.(col.lt.0)) go to 120
                    if((row.gt.127).or.(col.gt.127)) go to 120
                    dsum=(im(row,col)*coef(i,j))+dsum
120        continue
110        continue
            try(n,m)=dsum
90    continue
c  CALL SUBROUTINE TO INVERSE FILTER FILTERED IMAGE ARRAY
c      call_nonc(trv,xim)

```

## NONCAUSAL (CONT.)

```

c ADD THE MEAN OF THE INPUT IMAGE TO THE FILTERED RESULT
    do 145 i=0,rsize
        do 155 j=0,csize
            in(i,j)=try(i,j)+mean
155    continue
145    continue
c CREATE AN IMAGE DATA FILE
    call susintfile(im)
c SCALE THE RESULTING IMAGE ARRAY AND CONVERT TO BYTE FORM
    call scale(im,integ,max,min)
    call intbytel(integ,oin)
c WRITE THE GENERATED IMAGE INTO A FILE
    do 150 i=0,rsize
        do 160 j=0,csize
            a(ij)=oin(i,j)
160    continue
    write(2'i+1) (a(n),n=0,rsize)
150    continue
c CLOSE FILES
    close(unit=1)
    close(unit=2)
    close(3)
    stop
end

```

## CONV

```

c THIS PROGRAM GENERATES AN IMAGE TEXTURE USING A NONCAUSAL FIR FILTER
c WHOSE PARAMETERS ARE OBTAINED FROM A DATA FILE. THE FILTER IS
c APPLIED TO AN IMAGE. SUBROUTINES SCALE, INTBYTE, AND BYTEINT ARE
c USED TO PREPARE IMAGE DATA ARRAYS FOR DISPLAY.
c
c DEFINE VARIABLES
    byte a(0:127),oin(0:127,0:127)
    integer rsize,csize,i,j,rec,col,integ(0:127,0:127),fsize,mfsize
    real*8 tsun,in(0:127,0:127),coeff(-11:11,-11:11),msun,mean,oun(0:1
    :27,0:127)
c OPEN FILES
    open(unit=1,name='frathmann.intrap53.dat',type='old',access='dire
    lct',recordsize=32,maxrec=128)
    open(unit=2,name='frathmann.intrap59.dat',type='new',access='dire
    lct',recordsize=32,maxrec=128)
    open(3,file='frathmann.dataconv21.dat',status='old')
c DEFINE PARAMETERS
    rsize=127
    csize=127
    fsize=11
    mfsize=1+fsize
c READ FILTER PARAMETERS INTO AN ARRAY
    do 10 i=mfsize,fsize
        do 20 j=mfsize,fsize
            read(3,251) coeff(i,j)
25        format(120.12)
20        continue
10    continue
c READ IMAGE TO BE FILTERED INTO AN ARRAY AND CONVERT TO INTEGER
    do 30 i=0,rsize
        read(1'i+1) (a(n),n=0,127)
        do 40 j=0,csize
            bin(i,j)=a(j)
40    continue
30    continue
    call byteint(bin,integ)

```

## CONV (CONT.)

```

c CONVERT THE INTEGER ARRAY INTO A REAL ARRAY AND COMPUTE THE MEAN
  msum=0.0d0
  do 50 i=0,rsize
    do 50 j=0,csize
      in(i,j)=float(integ(i,j))
      msum=in(i,j)+msum
  50  continue
  50 continue
  mean=msum/float((rsize+1)*(csize+1))
  write(*,55)mean
  55 format(120.12)
c SUBTRACT THE MEAN FROM THE IMAGE ARRAY
  do 70 i=0,rsize
    do 80 j=0,csize
      in(i,j)=in(i,j)-mean
  80  continue
  70 continue
c FILTER THE IMAGE ARRAY
  do 90 n=0,rsize
    do 100 m=0,csize
      dsum=0.0d0
      do 110 i=nsize,fsize
        do 120 j=nsize,fsize
          row=n-i
          col=m-j
          if((row.lt.0).or.(col.lt.0)) go to 120
          if((row.gt.127).or.(col.gt.127)) go to 120
          dsum=(in(row,col)*coef(i,j))+dsum
  120  continue
  110  continue
      dsum(d,n)=dsum
  100 continue
  90 continue
c SCALE THE RESULTING IMAGE ARRAY AND CONVERT TO BYTE FORM
  call scale(huno,integ)
  call intovte(integ,21)
c WRITE THE GENERATED IMAGE INTO A FILE
  do 150 i=0,rsize
    do 160 j=0,csize
      a(j)=bitm(i,j)
  160  continue
      write(2*i+1) (a(n),n=0,rsize)
  150 continue
c CLOSE FILES
  close(unit=1)
  close(unit=2)
  close(3)
  stop
end

```

## SPECOR2

```
C THIS PROGRAM SOLVES EQUATIONS FOR SPECTRAL CONTENT AND CORRELATION
C IN ONE DIRECTION OF A GIVEN AUTOREGRESSIVE IMAGE MODEL. IT WRITES
C THESE RESULTS TO DEVICES 3 AND 4 RESPECTIVELY.
C DEFINE VARIABLES
    INTEGER I,K
    REAL=8 PI,ALPHA,THETA,W,ALPHAS,SXW,RXX(-49:49),Z,ALPHAK
C DEFINE PARAMETERS
    PI=3.141592654
    ALPHA=0.999999
    ALPHAS=ALPHA**2
C START X AXIS LOOP AND DEFINE X VALUES
    DO 10 I=0,99
        W=(-1.0*PI)+(2.0*PI*(FLOAT(I)/99.0))
C DO SPECTRAL ANALYSIS
        SXW=1.0/(1.0-(2.0*ALPHA*COS(W))+ALPHAS)
        WRITE(3,15)W,SXW
15     FORMAT(F10.5,1X,F10.5)
10    CONTINUE
C DO CORRELATION FUNCTION SOLUTION
    DO 20 K=0,49
        ALPHAK=ALPHA**K
        RXX(K)=(ALPHAK/(1.0-ALPHAS))
        I=-1+K
        RXX(I)=RXX(K)
20    CONTINUE
    DO 30 K=-49,49
        Z=FLOAT(K)
        WRITE(4,25)Z,RXX(K)
25    FORMAT(F10.0,1X,F10.15)
30    CONTINUE
    STOP
    END
```

# SPECOR3

```

C THIS PROGRAM SOLVES EQUATIONS FOR SPECTRAL CONTENT AND CORRELATION
C IN ONE DIRECTION OF A GIVEN AUTOREGRESSIVE IMAGE MODEL. IT WRITES
C THESE RESULTS TO DEVICES 3 AND 4 RESPECTIVELY.
C DEFINE VARIABLES
    INTEGER I,J,M
    REAL=8 PI,ALPHA,TMTHA,ALPHAS,ALSPA,ALPHAG,TWOTH,COS2T,W,A,B
    =,COS2W,SXW1,SXW,K,ALPHAK,KTHETA,ALPH2K,KP2TMP,KTMP,INTERM,RXK(-49:
    *49),Z
C DEFINE PARAMETERS
    PI=3.141592654
    ALPHA=0.9
    THETA=0.0=(PI/I2.0)
    IF(THETA.EQ.0.0) THETA=0.000001
    ALPHAS=ALPHA**3
    ALSPA=ALPHAS+ALPHA
    ALPHAS=ALPHA**2
    ALPHAG=ALPHA**4
    COS2T=COS(2.0*THETA)
    TWOTH=2.0*THETA
C START X AXIS LOOP AND DEFINE X VALUES
C DO 10 I=0,99
C     W=(-1.0*PI)+(2.0*PI)=(FLOAT(I)/99.0)
C DO SPECTRAL ANALYSIS
C     A=THETA-W
C     B=THETA+W
C     COS2W=COS(2.0*W)
C     SXW1=ALPHAS-(COS(A)=ALSPA)*(ALPHAS=COS2T)-(COS(B)=ALSPA)*(ALPHAS
C     ==COS2W)
C     SXW=1.0/(I.0+ALPHAG*(2.0=SXW1))
C     WRITE(3,15)W,SXW
C 15  FORMAT(F10.5,1X,F10.5)
C 10 CONTINUE
C DO CORRELATION FUNCTION SOLUTION
    DO 30 J=0,49
        K=FLOAT(J)
        ALPHAK=ALPHA**J
        KTHETA=K*THETA
        KP2TMP=((2.0*K)=THETA)-PI
        KTMP=KTHETA-PI
        INTERM=(COS(KP2TMP)-(ALPHAS=COS(KTMP)))/(1.0+ALPHAG-(2.0*ALPHA
        *S=COS(TWOTH)))
        RXK(J)=(ALPHAK*(2.0*(SIN(THETA)**2)))*((COS(KTHETA)/(1.0-ALPHA
        *S))+INTERM)
        GO TO 36
C 35  RXK(J)=(ALPHAK*(2.0*(K**2)))/(2.0*((1.0-ALPHAS)**2))
C 36  M=-1=J
        RXK(M)=RXK(J)
C 30 CONTINUE
    DO 40 J=-49,49
        Z=FLCAT(J)
        WRITE(4,45)Z,RXK(J)
C 45  FORMAT(F10.1,1X,F30.15)
C 40 CONTINUE
        STOP
    END

```

# SPECOR3A

```

C THIS PROGRAM SOLVES EQUATIONS FOR SPECTRAL CONTENT AND CORRELATION
C IN ONE DIRECTION OF A GIVEN AUTOREGRESSIVE IMAGE MODEL. IT WRITES
C THESE RESULTS TO DEVICES 3 AND 4 RESPECTIVELY.
C DEFINE VARIABLES
    INTEGER I,J,M
    REAL=8 PI,A,B,AB,AMBS,AS,BS,AK1,AK2,BK1,BK2,INTERM,RXK(-49:49),Z,A
    #K,KP2,KP1,OPAS,OMAS
C DEFINE PARAMETERS
    PI=3.141592654
    A=-0.9
    B=-0.89999
    AB=A*B
    AMBS=(A-B)**2
    AS=A**2
    BS=B**2
C START X AXIS LOOP AND DEFINE X VALUES
    DO 10 I=0,99
        W=(-1.0*PI)*(2.0*PI*(FLOAT(I)/99.0))
C DO SPECTRAL ANALYSIS
    COSW=COS(2.0*W)
    SXW1=1.0-(2.0*(A+B*(AS=B)+(A=BS))*COS(W))+(2.0*A*B=COS2W)
    SXW=1.0/(SXW1)*AS+(2.0*A=B)*BS*(AS=BS))
    WRITE(3,IS)W,SXW
15   FORMAT(F10.5,IX,F10.5)
10 CONTINUE
C DO CORRELATION FUNCTION SOLUTION
    DO 30 J=0,49
        IF(A.EQ.B) GO TO 35
        AK1=A**((J+1))
        AK2=A**((J+2))
        BK1=B**((J+1))
        BK2=B**((J+2))
        INTERM=(AK2/(1.0-AS)-(((AK1=B)*A=BK1))/(1.0-AB))+(BK2/(1.0-BS))
        *)
        RXK(J)=(1.0/AMBS)=INTERM
        GO TO 36
35   AK=A**J
        KP2=FLOAT(J+2)
        KP1=FLOAT(J+1)
        OPAS=1.0*AS
        OMAS=1.0-AS
        RXK(J)=((KP2=KP1=AK)/(2.0*OMAS))-((FLOAT(J)=KP1=AK)/(2.0*OPAS))
        *)
36   M=-1=J
        RXK(M)=RXK(J)
30 CONTINUE
    DO 40 J=-49,49
        Z=FLOAT(J)
        WRITE(4,45)Z,RXK(J)
45   FORMAT(F10.1,IX,F30.15)
40 CONTINUE
    STOP
    END

```

## PIECE

```
c THIS PROGRAM TAKES A 128X128 PIECE OUT OF A 512X512 IMAGE DATA FILE
c AND PLACES THAT DATA INTO AN IMAGE FILE
c
c
c
c DEFINE VARIABLES
  byte a(0:511),dim(0:511,0:511),ao(0:127)
c OPEN FILES
  open(unit=1,name='[ pathname ]trees.dat',type='old',access='direct'
       ;recsize=32,maxrec=128)
  open(unit=2,name='[ pathname ]trees64.dat',type='new',access='direct'
       ;recsize=16,maxrec=64)
c READ IMAGE TO BE FILTERED INTO AN ARRAY
  do 30 i=0,127
    read(1'i+1) (a(n),n=0,127)
    do 40 j=0,127
      b(i,j)=a(j)
    40 continue
  30 continue
c WRITE THE DESIRED IMAGE PORTION INTO A FILE
  do 150 i=63,126
    do 160 j=i-63
      k=j-63
      ao(k)=dim(i,j)
    160 continue
    write(2'i-62) (ao(n),n=0,63)
  150 continue
c CLOSE FILES
  close(unit=1)
  close(unit=2)
  stop
end
```

## INTFILE

```
c THIS PROGRAM CREATES A 128X128 OR 64X64 INTEGER FILE FROM A 128X128
c INPUT IMAGE FILE
c
c
c
c DEFINE VARIABLES
  byte a(0:127),dim(0:127,0:127)
  integer i,j,n,k,l,inteo(0:127,0:127),sum,high,low
  real**8 mean
c OPEN FILES
  open(unit=1,file='[ pathname ]filter2.dat',type='old',access='direct'
       ;recsize=32,maxrec=128)
  open(unit=2,file='[ pathname ]final.dat',status='new')
c READ IMAGE TO BE FILTERED INTO AN ARRAY
  do 30 i=0,127
    read(1'i+1) (a(n),n=0,127)
    do 40 j=0,127
      b(i,j)=a(j)
    40 continue
  30 continue
c CONVERT THE BYTE ARRAY INTO AN INTEGER ARRAY
  call byteint(nin,inteo)
```

## INTFILE (CONT.)

```

c COMPUTE HIGH AND LOW VALUES OF THE IMAGE DATA AND WRITE TO TERMINAL
high=0
low=255
do 31 i=0,127
    do 32 j=0,127
        if(intea(i,j).lt.low) low=intea(i,j)
        if(intea(i,j).gt.high) high=intea(i,j)
32    continue
31    continue
    write(*,33)high,low
33 format(2i0)
c COMPUTE MEAN OF INTEGER ARRAY AND SUBTRACT IT FROM THE DATA
c sum=0
c   to 35 i=0,127
c     do 45 j=0,127
c       sum=sum+intea(i,j)
c   45  continue
c   35 continue
c   mean=float(sum)/(129.0*129.0)
c   write(*,34)sum,mean
c   34 format(i10,e10.3)
c   do 35 i=0,127
c     do 45 j=0,127
c       inten(i,j)=intea(i,j)-int(mean+0.5)
c   45  continue
c   35 continue
c WRITE INTEGER ARRAY INTO A 128X128 DATA FILE
c   k=124
c   l=128
c   write(2,32)k,l
c   32 format(2i5)
c   do 50 i=0,127
c     do 60 j=0,127,15
c       write(2,70)intea(i,j),intea(i,j+1),intea(i,j+2),intea(i,j+3),i
c       *ntea(i,j+4),intea(i,j+5),intea(i,j+6),intea(i,j+7),intea(i,j+8),i
c       *ntea(i,j+9),intea(i,j+10),intea(i,j+11),intea(i,j+12),intea(i,j+13)
c       *,intea(i,j+14),intea(i,j+15)
c   70    format(1si5)
c   60  continue
c   50 continue
c CLOSE FILE
close(unit=1)
close(unit=2)
stop
end

```

## TRANS

```

c THIS PROGRAM READS DATA OUT OF A FILE IN FREE FORMAT AND CONVERTS
c IT TO A PROGRAM READABLE FORMAT.
integer i,j
real*8 a(16)
open(1,file='rathmann.dat1adwat.dat',status='old')
open(2,file='rathmann.dat1fcnwat.dat',status='new')
read(1,*)
  i,j,a(1),a(2),a(3),a(4),a(5),a(6),a(7),a(8),a(9),a(10),a(11),
  a(12),a(13),a(14),a(15),a(16)
  do 10 k=1,16
    write(2,20)a(k)
20    format(d20.12)
10  continue
stop
end

```

## NSHP

```

c THIS PROGRAM GENERATES AN IMAGE TEXTURE USING WHITE NOISE AS AN INPUT
c TO AN AUTOREGRESSIVE FILTER. THESE PARAMETERS ARE OBTAINED FROM THE
c FILE FILDEF. SUBROUTINE PGAUSS IS USED TO GENERATE THE INPUT WHITE
c NOISE AND SUBROUTINES SCALE AND INTBYTE ARE USED TO PREPARE IMAGE
c DATA ARRAYS FOR DISPLAY.
c
c DEFINE VARIABLES
      bvtl_a(0:127),dim(0:127,0:127)
      integer n,seed,rsize,csize,om1,nm1,i,ipl,row,col,integ(0:127,0:1
      :27),nm1
      real*8 val1,val2,w(0:127,0:127),tsum,in(0:127,0:127),coef(-2:9,-2
      ::9)
c OPEN FILES
      open(unit=1,name='frathmann.datalen.dat',type='new',access='direct
      ',recsize=32,maxrec=128)
      open(unit=2,name='frathmann.datarotate.dat',type='new',access='di
      rect',recsize=32,maxrec=128)
      open(3,file='frathmann.datafc3a.dat',status='old')
      open(4,file='frathmann.datalts.dat',status='new')
c DEFINE PARAMETERS
      seed=1234567
      rsize=127
      csize=127
      om1=2
      nm1=4
      nm1=-1+om1
c CREATE WHITE NOISE ARRAY
      do 10 i=0,rsize
         do 20 j=0,csize-1
            jpl=j+1
            call pgauss(seed,val1,val2)
            w(i,j)=val1
            w(i,jpl)=val2
20      continue
10      continue
c SCALE ARRAY W AND CONVERT TO BYTE FORM
      call scale(w,integ)
      call intbyte(integ,oin)
c WRITE THE WHITE NOISE IMAGE ARRAY TO A FILE
      do 30 i=0,rsize
         do 40 j=0,csize
            a(j)=oin(i,j)
40      continue
            write(l'i+1') (a(n),n=0,rsize)
30      continue
c READ FILTER PARAMETERS INTO AN ARRAY
      do 50 i=nm1,nm1
         do 60 j=0,csize
            read(3,55) coef(i,j)
55      format(720.12)
60      continue
50      continue
c APPLY WHITE NOISE TO THE AUTOREGRESSIVE FILTER
      do 110 n=0,csize
         do 120 m=0,rsize
            dsun=0.0d0
            do 130 i=nm1,nm1
               do 140 j=0,csize
                  if((i.eq.0).and.(j.eq.0)) do to 140
                  row=i
                  col=j
                  if((row.lt.0).or.(col.lt.0)) do to 140
                  dsun=(coef(i,j)*in(row,col))+tsum
140             continue
130             continue
               in(n,m)=(-1.0d0*dsun)+w(n,m)
120             continue
110             continue

```

## NSHP (CONT.)

```

c SCALE THE RESULTING IMAGE ARRAY AND CONVERT TO BYTE FORM
    call scale(arr,integ)
    call int2byte(integ,im)
c WRITE THE GENERATED IMAGE INTO A FILE
    do 150 i=0,rsize
        do 160 j=0,csize
            a(j)=im(i,j)
160    continue
        write(2,'i1') (a(n),n=0,rsize)
150 continue
c CLOSE FILES
    close(unit=1)
    close(unit=2)
    close(3)
c close(4)
stop
end

```

## PGAUSS

```

SUBROUTINE PGAUSS(k,z1,z2)
REAL*8 A,B,Z1,Z2
INTEGER RND K
A = SQRT(-2.0D0 * ALOG(RAN(K)))
B = 5.283195D0 * RAN(K)
Z1 = A * COS(B)
Z2 = A * SIN(B)
RETURN
END

```

## SCALE

```

SUBROUTINE scale(arr,integ,max,min)
c THIS SUBROUTINE SCALES AN ARRAY TO INTEGER VALUES BETWEEN A GIVEN
c MAXIMUM AND A MINIMUM
c DEFINE VARIABLES
    integer integ(0:127,0:127),i,j
    real*8 arr(0:127,0:127),high,low,mean,max,min
c PERFORM SCALING
    max=-170.0,-0.04
    low=100.0,0.04
    do 10 i=0,127
        do 20 j=0,127
            if(arr(i,j).le.low) low=arr(i,j)
            if(arr(i,j).ge.max) max=arr(i,j)
            sum=arr(i,j)+sum
20    continue
10    continue
    mean=sum/(128.0*128.0)
    arr(:,25)=high,low,mean
25    format(' prescaled image data= high=',d12.5,' low=',d12.5,' mean='
     ,d12.5)
c CONTINUE SCALING AND CONVERT TO INTEGER FORM
    do 30 i=0,127
        do 40 j=0,127
            integ(i,j)=int(((arr(i,j)-low)*((max-min)/(high-low)))+0.5d0)
            integ(i,j)=integ(i,j)+int(min)
40    continue
30    continue

```

## SCALE (CONT.)

```

c CALCULATE HIGH, LOW, AND MEAN OF SCALED IMAGE
sum=0.0
high=-10000.0
low=10000.0
do 50 i=0,127
    do 50 j=0,127
        if(float(integ(i,j)).gt.high) high=float(integ(i,j))
        if(float(integ(i,j)).lt.low) low=float(integ(i,j))
        sum=float(integ(i,j))+sum
50 continue
50 continue
mean=sum/(128.0*128.0)
write(*,65)high,low,mean
65 format(' scaled image data- high=',d12.5,' low=',d12.5,' mean=',d1
2.5)
return
end

```

## INTBYTE

```

      subroutine intbyte(integ,i5)
c THIS SUBROUTINE TAKES AN INTEGER ARRAY AND CONVERTS IT INTO A
c      BYTE ARRAY
c DEFINE VARIABLES
      integer i,j,integ(0:127,0:127),n
      byte bin(0:127,0:127)
c PERFORM CONVERSION
      n=0
      do 10 i=0,127
          do 20 j=0,127
              if((integ(i,j).lt.0).or.(integ(i,j).gt.255)) n=n+1
              if(integ(i,j).lt.0) bin(i,j)=128
              if(integ(i,j).gt.255) bin(i,j)=127
              if((integ(i,j).le.127).and.(integ(i,j).ge.0)) bin(i,j)=
              integ(i,j)
              if((integ(i,j).gt.127).and.(integ(i,j).le.255)) bin(i,j)=
              integ(i,j)-256
20      continue
10      continue
      write(*,30) n
30      format(' THE NUMBER OF POINTS OUT OF RANGE IS',i5)
      return
end

```

## BYTEINT

```

      SUBROUTINE BYTEINT(aint,inteo)
C THIS PROGRAM TAKES A BYTE ARRAY AND CONVERTS IT INTO AN INTEGER ARRAY
C
C DEFINE VARIABLES
      integer inteo(0:127,0:127),int
      byte bin(0:127,0:127)
C PERFORM CONVERSION
      n=0
      do 10 i=0,127
         do 20 j=0,127
            if((bin(i,j).lt.-128).or.(bin(i,j).gt.127)) n=n+1
            if(bin(i,j).lt.-128) inteo(i,j)=-128
            if(bin(i,j).gt.127) inteo(i,j)=127
            if((bin(i,j).ge.-128).and.(bin(i,j).lt.0)) inteo(i,j)=
              bin(i,j)+256
            if((bin(i,j).ge.0).and.(bin(i,j).le.127)) inteo(i,j)=bin(i,j)
20      continue
10      continue
      write(*,30)n
30      format(' THE NUMBER OF POINTS OUT OF RANGE IS',IS)
      return
      end

```

## SUBINTFILE

```

      SUBROUTINE SUBINTFILE()
C THIS SUBROUTINE CREATES A 128X128 INTEGER DATA FILE FROM FILTERED
C IMAGE DATA
C
C DEFINE VARIABLES
      integer i,j,n,x,l,inteo(0:127,0:127),sum
      real*4 mean,in(0:127,0:127),high,low
C OPEN FILES
      open(l,file='fathmann.dat',status='new')
C COMPUTE HIGH AND LOW VALUES OF THE IMAGE DATA AND WRITE TO TERMINAL
      high=-1.045
      low=1.045
      do 31 i=0,127
         do 32 j=0,127
            if(im(i,j).lt.low) low=im(i,j)
            if(im(i,j).gt.high) high=im(i,j)
32      continue
31      continue
      write(*,33)high,low
33      format(2F20.12)
C COMPUTE MEAN OF INTEGER ARRAY AND SUBTRACT IT FROM THE DATA
C
      sum=0
      do 35 i=0,127
         do 45 j=0,127
            sum=sum+inteo(i,j)
45      continue
35      continue
      mean=float(sum)/(128.0*128.0)
      write(*,34)sum,mean
34      format(i10,e10.3)
      do 35 i=0,127
         do 45 j=0,127
            inteo(i,j)=inteo(i,j)-int(mean+0.5)
45      continue
35      continue

```

## SUBINTFILE

```

c CONVERT IMAGE DATA INTO INTEGER FORM
  do 45 i=0,127
    do 55 j=0,127
      inteq(i,j)=int(im(i,j)+0.5d0)
  55 continue
  45 continue
c WRITE INTEGER ARRAY INTO A 128X128 DATA FILE
  k=128
  l=128
  write(1,39)k,l
  39 format(2i5)
  do 50 i=0,127
    do 60 j=0,112,16
      write(1,77)inteq(i,j),inteq(i,i+1),inteq(i,i+2),inteq(i,i+3),
      inteq(i,i+4),inteq(i,i+5),inteq(i,i+6),inteq(i,i+7),inteq(i,i+8),in-
      teq(i,i+9),inteq(i,i+10),inteq(i,i+11),inteq(i,i+12),inteq(i,i+13),
      *inteq(i,i+14),inteq(i,i+15)
  70   format(16i5)
  60   continue
  50 continue
c CLOSE FILES
  close(unit=1)
  return
end

```

## NONC

```

subroutine nonc(trv,im)
c THIS SUBROUTINE TAKES AN IMAGE ARRAY AND FILTERS IT WITH A
c VONCAUSA_ FIR FILTER.
c
c DEFINE VARIABLES
  integer nsrsize,cszie,i,j,row,col,fszie,mfsize,index1,index2
  real*4 tsum,im(0:127,0:127),coef(-10:10,-10:10),nsun,mean,sums(-1:
  ;128,-1:128),high,low,trv(0:127,0:127)
c OPEN FILES
  open(1,file='frathmann.datalncfile21.dat',status='old')
c DEFINE PARAMETERS
  index1=0
  index2=127
  nsrsize=127
  cszie=127
  fszie=10
  mfsize=1+fszie
c READ FILTER PARAMETERS INTO AN ARRAY
  do 10 i=mfsize,fszie
    do 20 j=mfsize,fszie
      read(1,25) coef(i,j)
  25   format(320.12)
  20   continue
  10 continue
c TAKE THE INPUT ARRAY AND COMPUTE THE MEAN, HIGH, AND LOW VALUES
  nsun=0.0d0
  high=-10000.0d0
  low=10000.0d0
  do 50 i=index1,index2
    do 60 j=index1,index2
      tsum=trv(i,j)+tsum
      if(trv(i,j).gt.high) high=trv(i,j)
      if(trv(i,j).lt.low) low=trv(i,j)
  60   continue
  50 continue
  mean=tsum/float((index2-index1+1)*(index2-index1+1))
  write(*,55)mean,high,low
  55 format(' inout image array- mean=',d12.5,' high=',d12.5,' low=',d1
  *2.5)

```

## NONC (CONT.)

```
c FILTER THE IMAGE ARRAY
  do 90 n=index1,index2
  do 100 m=index1,index2
    dsun=0.010
    do 110 i=nfsize,fsize
      do 120 j=nfsize,fsize
        row=i-1
        col=j-1
        if((row.lt.index1).or.(col.lt.index1)) go to 120
        if((row.gt.index2).or.(col.gt.index2)) go to 120
        dsun=(trev(row,col)*coef(i,j))+dsun
120    continue
110    continue
    im(r,n)=dsun
100   continue
90  continue
c CONVERT THE 130X130 ARRAY INTO A 128X128 ARRAY, IF NECESSARY
c   do 130 i=0,127
c     do 140 j=0,127
c       im(i,j)=dunb(i,j)
c 140   continue
c 130 continue
  close(1)
  return
end
```

# MAKMAT

```

      D E+MAKMAT.DIG:A,C,D,E,F,G,H,I,J,K,L,M,O
      D1=0
      A2=1 11 66 80
      AF0:0,01+4
      AF0:0,11+4
      DUM=0
      I=0
      START1=0
      START2=1
      LOOP1: I=I+1
      START1=START1+(I-1)
      START2=START2+(I+1)
      ROW=1
      LOOP2: ROW=ROW+1
      →(ROW=0)/ROW1
      →(ROW=I)/ROWI
      COL1=COL1+1
      COL2=COL1+(I-1)
      COL3=COL2+1
      COL4=COL3+1
      COL5=COL4+1
      AF1:ROW,COL1,I+1
      AF1:ROW,COL2,I+1
      AF1:ROW,COL3,I+4
      AF1:ROW,COL4,I+1
      AF1:I=100/LOOP1
      AF1:ROW,COL5,I+1
      →(ROW<I)/LOOP2
      ROW, COL1=START1
      COL2=COL1+1
      COL3=COL2+1
      COL4=COL3+1
      AF1:ROW,COL1,I+1
      AF1:ROW,COL2,I+4
      AF1:ROW,COL3,I+2
      →I=100/LOOP2
      AF1:ROW,COL4,I+1
      →I=100/LOOP2
      ROW, COL1=START2
      COL2=COL1+1
      COL3=COL2+(I+1)
      AF1:ROW,COL2,I+2
      AF1:ROW,COL3,I+4
      AF1:I=100/DONE
      AF1:ROW,COL3,I+2
      DONE=I=100 ↓AF0,1
      D=100 ↓AF1,1
      D=100 ↓AF2,1
      D=100 ↓AF3,1
      D=100 ↓AF4,1
      H=100 ↓AF5,1
      K=100 ↓AF6,1
      L=100 ↓AF7,1
      N=100 ↓AF8,1
      O=100 ↓AF9,1
      P=100 ↓AF10,1
      Q=100 ↓AF11,1
      R=100 ↓AF12,1
      S=100 ↓AF13,1
      T=100 ↓AF14,1
      U=100 ↓AF15,1
      V=100 ↓AF16,1
      W=100 ↓AF17,1
      X=100 ↓AF18,1
      Y=100 ↓AF19,1
      Z=100 ↓AF20,1
      ↓AF0,1

```

## CC2

```

1   F Y+X CC2 H,I1,I2,O10,N1,N2,T1,T2
2   A FUNCTION TO DO 2-D CIRCULAR CONVOLUTION
3   A ((I1+I2)/N1) * O10 = X1/REGIN
4   D= 'IMPROPER ARGUMENTS'
5   D0
6   EGIN O10=0
7   N1+1*I1
8   N2+1*I2
9   Y=(N2,N1)=0
10  T1=0
11  LOOP1 T1+=(-(I1+1))*PHH
12  T2=0
13  LOOP2 T2+=(-(I2+1))*O10
14  Y(I2,I1)=Y/I2
15  ?(N2,I2,I1)=T2
16  ?(N1,I1)=T1
17  END

```

## GETDATA

```

1   F GETDATA NAME,I1,I2,O10,N1,N2,T1,T2
2   A FUNCTION TO READ FREE-COBOLATED DATA (VECTOR, ARRAYS, ETC.) FROM DISK FILE
3   D FILENAME, I1,I2,O10,N1,N2,T1,T2
4   D RECORDS,REC
5   D SYSPRINT,SYSPUNCH,ERRCODE
6   D INFILE,OUTFILE,LINE
7   D RECNO,RECNO1
8   D RECNO2
9   D RECNO3
10  D RECNO4
11  D RECNO5
12  D RECNO6
13  D RECNO7
14  D RECNO8
15  D RECNO9
16  D RECNO10
17  D RECNO11
18  D RECNO12
19  D RECNO13
20  D RECNO14
21  D RECNO15
22  D RECNO16
23  D RECNO17
24  D RECNO18
25  D RECNO19
26  D RECNO20
27  D RECNO21
28  D RECNO22
29  D RECNO23
30  D RECNO24
31  D RECNO25
32  D RECNO26
33  D RECNO27
34  D RECNO28
35  D RECNO29
36  D RECNO30
37  D RECNO31
38  D RECNO32
39  D RECNO33
40  D RECNO34
41  D RECNO35
42  D RECNO36
43  D RECNO37
44  D RECNO38
45  D RECNO39
46  D RECNO40
47  D RECNO41
48  D RECNO42
49  D RECNO43
50  D RECNO44
51  D RECNO45
52  D RECNO46
53  D RECNO47
54  D RECNO48
55  D RECNO49
56  D RECNO50
57  D RECNO51
58  D RECNO52
59  D RECNO53
60  D RECNO54
61  D RECNO55
62  D RECNO56
63  D RECNO57
64  D RECNO58
65  D RECNO59
66  D RECNO60
67  D RECNO61
68  D RECNO62
69  D RECNO63
70  D RECNO64
71  D RECNO65
72  D RECNO66
73  D RECNO67
74  D RECNO68
75  D RECNO69
76  D RECNO70
77  D RECNO71
78  D RECNO72
79  D RECNO73
80  D RECNO74
81  D RECNO75
82  D RECNO76
83  D RECNO77
84  D RECNO78
85  D RECNO79
86  D RECNO80
87  D RECNO81
88  D RECNO82
89  D RECNO83
90  D RECNO84
91  D RECNO85
92  D RECNO86
93  D RECNO87
94  D RECNO88
95  D RECNO89
96  D RECNO90
97  D RECNO91
98  D RECNO92
99  D RECNO93
100  D RECNO94
101  D RECNO95
102  D RECNO96
103  D RECNO97
104  D RECNO98
105  D RECNO99
106  D RECNO100
107  D RECNO101
108  D RECNO102
109  D RECNO103
110  D RECNO104
111  D RECNO105
112  D RECNO106
113  D RECNO107
114  D RECNO108
115  D RECNO109
116  D RECNO110
117  D RECNO111
118  D RECNO112
119  D RECNO113
120  D RECNO114
121  D RECNO115
122  D RECNO116
123  D RECNO117
124  D RECNO118
125  D RECNO119
126  D RECNO120
127  D RECNO121
128  D RECNO122
129  D RECNO123
130  D RECNO124
131  D RECNO125
132  D RECNO126
133  D RECNO127
134  D RECNO128
135  D RECNO129
136  D RECNO130
137  D RECNO131
138  D RECNO132
139  D RECNO133
140  D RECNO134
141  D RECNO135
142  D RECNO136
143  D RECNO137
144  D RECNO138
145  D RECNO139
146  D RECNO140
147  D RECNO141
148  D RECNO142
149  D RECNO143
150  D RECNO144
151  D RECNO145
152  D RECNO146
153  D RECNO147
154  D RECNO148
155  D RECNO149
156  D RECNO150
157  D RECNO151
158  D RECNO152
159  D RECNO153
160  D RECNO154
161  D RECNO155
162  D RECNO156
163  D RECNO157
164  D RECNO158
165  D RECNO159
166  D RECNO160
167  D RECNO161
168  D RECNO162
169  D RECNO163
170  D RECNO164
171  D RECNO165
172  D RECNO166
173  D RECNO167
174  D RECNO168
175  D RECNO169
176  D RECNO170
177  D RECNO171
178  D RECNO172
179  D RECNO173
180  D RECNO174
181  D RECNO175
182  D RECNO176
183  D RECNO177
184  D RECNO178
185  D RECNO179
186  D RECNO180
187  D RECNO181
188  D RECNO182
189  D RECNO183
190  D RECNO184
191  D RECNO185
192  D RECNO186
193  D RECNO187
194  D RECNO188
195  D RECNO189
196  D RECNO190
197  D RECNO191
198  D RECNO192
199  D RECNO193
200  D RECNO194
201  D RECNO195
202  D RECNO196
203  D RECNO197
204  D RECNO198
205  D RECNO199
206  D RECNO200
207  D RECNO201
208  D RECNO202
209  D RECNO203
210  D RECNO204
211  D RECNO205
212  D RECNO206
213  D RECNO207
214  D RECNO208
215  D RECNO209
216  D RECNO210
217  D RECNO211
218  D RECNO212
219  D RECNO213
220  D RECNO214
221  D RECNO215
222  D RECNO216
223  D RECNO217
224  D RECNO218
225  D RECNO219
226  D RECNO220
227  D RECNO221
228  D RECNO222
229  D RECNO223
230  D RECNO224
231  D RECNO225
232  D RECNO226
233  D RECNO227
234  D RECNO228
235  D RECNO229
236  D RECNO230
237  D RECNO231
238  D RECNO232
239  D RECNO233
240  D RECNO234
241  D RECNO235
242  D RECNO236
243  D RECNO237
244  D RECNO238
245  D RECNO239
246  D RECNO240
247  D RECNO241
248  D RECNO242
249  D RECNO243
250  D RECNO244
251  D RECNO245
252  D RECNO246
253  D RECNO247
254  D RECNO248
255  D RECNO249
256  D RECNO250
257  D RECNO251
258  D RECNO252
259  D RECNO253
260  D RECNO254
261  D RECNO255
262  D RECNO256
263  D RECNO257
264  D RECNO258
265  D RECNO259
266  D RECNO260
267  D RECNO261
268  D RECNO262
269  D RECNO263
270  D RECNO264
271  D RECNO265
272  D RECNO266
273  D RECNO267
274  D RECNO268
275  D RECNO269
276  D RECNO270
277  D RECNO271
278  D RECNO272
279  D RECNO273
280  D RECNO274
281  D RECNO275
282  D RECNO276
283  D RECNO277
284  D RECNO278
285  D RECNO279
286  D RECNO280
287  D RECNO281
288  D RECNO282
289  D RECNO283
290  D RECNO284
291  D RECNO285
292  D RECNO286
293  D RECNO287
294  D RECNO288
295  D RECNO289
296  D RECNO290
297  D RECNO291
298  D RECNO292
299  D RECNO293
300  D RECNO294
301  D RECNO295
302  D RECNO296
303  D RECNO297
304  D RECNO298
305  D RECNO299
306  D RECNO300
307  D RECNO301
308  D RECNO302
309  D RECNO303
310  D RECNO304
311  D RECNO305
312  D RECNO306
313  D RECNO307
314  D RECNO308
315  D RECNO309
316  D RECNO310
317  D RECNO311
318  D RECNO312
319  D RECNO313
320  D RECNO314
321  D RECNO315
322  D RECNO316
323  D RECNO317
324  D RECNO318
325  D RECNO319
326  D RECNO320
327  D RECNO321
328  D RECNO322
329  D RECNO323
330  D RECNO324
331  D RECNO325
332  D RECNO326
333  D RECNO327
334  D RECNO328
335  D RECNO329
336  D RECNO330
337  D RECNO331
338  D RECNO332
339  D RECNO333
340  D RECNO334
341  D RECNO335
342  D RECNO336
343  D RECNO337
344  D RECNO338
345  D RECNO339
346  D RECNO340
347  D RECNO341
348  D RECNO342
349  D RECNO343
350  D RECNO344
351  D RECNO345
352  D RECNO346
353  D RECNO347
354  D RECNO348
355  D RECNO349
356  D RECNO350
357  D RECNO351
358  D RECNO352
359  D RECNO353
360  D RECNO354
361  D RECNO355
362  D RECNO356
363  D RECNO357
364  D RECNO358
365  D RECNO359
366  D RECNO360
367  D RECNO361
368  D RECNO362
369  D RECNO363
370  D RECNO364
371  D RECNO365
372  D RECNO366
373  D RECNO367
374  D RECNO368
375  D RECNO369
376  D RECNO370
377  D RECNO371
378  D RECNO372
379  D RECNO373
380  D RECNO374
381  D RECNO375
382  D RECNO376
383  D RECNO377
384  D RECNO378
385  D RECNO379
386  D RECNO380
387  D RECNO381
388  D RECNO382
389  D RECNO383
390  D RECNO384
391  D RECNO385
392  D RECNO386
393  D RECNO387
394  D RECNO388
395  D RECNO389
396  D RECNO390
397  D RECNO391
398  D RECNO392
399  D RECNO393
400  D RECNO394
401  D RECNO395
402  D RECNO396
403  D RECNO397
404  D RECNO398
405  D RECNO399
406  D RECNO400
407  D RECNO401
408  D RECNO402
409  D RECNO403
410  D RECNO404
411  D RECNO405
412  D RECNO406
413  D RECNO407
414  D RECNO408
415  D RECNO409
416  D RECNO410
417  D RECNO411
418  D RECNO412
419  D RECNO413
420  D RECNO414
421  D RECNO415
422  D RECNO416
423  D RECNO417
424  D RECNO418
425  D RECNO419
426  D RECNO420
427  D RECNO421
428  D RECNO422
429  D RECNO423
430  D RECNO424
431  D RECNO425
432  D RECNO426
433  D RECNO427
434  D RECNO428
435  D RECNO429
436  D RECNO430
437  D RECNO431
438  D RECNO432
439  D RECNO433
440  D RECNO434
441  D RECNO435
442  D RECNO436
443  D RECNO437
444  D RECNO438
445  D RECNO439
446  D RECNO440
447  D RECNO441
448  D RECNO442
449  D RECNO443
450  D RECNO444
451  D RECNO445
452  D RECNO446
453  D RECNO447
454  D RECNO448
455  D RECNO449
456  D RECNO450
457  D RECNO451
458  D RECNO452
459  D RECNO453
460  D RECNO454
461  D RECNO455
462  D RECNO456
463  D RECNO457
464  D RECNO458
465  D RECNO459
466  D RECNO460
467  D RECNO461
468  D RECNO462
469  D RECNO463
470  D RECNO464
471  D RECNO465
472  D RECNO466
473  D RECNO467
474  D RECNO468
475  D RECNO469
476  D RECNO470
477  D RECNO471
478  D RECNO472
479  D RECNO473
480  D RECNO474
481  D RECNO475
482  D RECNO476
483  D RECNO477
484  D RECNO478
485  D RECNO479
486  D RECNO480
487  D RECNO481
488  D RECNO482
489  D RECNO483
490  D RECNO484
491  D RECNO485
492  D RECNO486
493  D RECNO487
494  D RECNO488
495  D RECNO489
496  D RECNO490
497  D RECNO491
498  D RECNO492
499  D RECNO493
500  D RECNO494
501  D RECNO495
502  D RECNO496
503  D RECNO497
504  D RECNO498
505  D RECNO499
506  D RECNO500
507  D RECNO501
508  D RECNO502
509  D RECNO503
510  D RECNO504
511  D RECNO505
512  D RECNO506
513  D RECNO507
514  D RECNO508
515  D RECNO509
516  D RECNO510
517  D RECNO511
518  D RECNO512
519  D RECNO513
520  D RECNO514
521  D RECNO515
522  D RECNO516
523  D RECNO517
524  D RECNO518
525  D RECNO519
526  D RECNO520
527  D RECNO521
528  D RECNO522
529  D RECNO523
530  D RECNO524
531  D RECNO525
532  D RECNO526
533  D RECNO527
534  D RECNO528
535  D RECNO529
536  D RECNO530
537  D RECNO531
538  D RECNO532
539  D RECNO533
540  D RECNO534
541  D RECNO535
542  D RECNO536
543  D RECNO537
544  D RECNO538
545  D RECNO539
546  D RECNO540
547  D RECNO541
548  D RECNO542
549  D RECNO543
550  D RECNO544
551  D RECNO545
552  D RECNO546
553  D RECNO547
554  D RECNO548
555  D RECNO549
556  D RECNO550
557  D RECNO551
558  D RECNO552
559  D RECNO553
560  D RECNO554
561  D RECNO555
562  D RECNO556
563  D RECNO557
564  D RECNO558
565  D RECNO559
566  D RECNO560
567  D RECNO561
568  D RECNO562
569  D RECNO563
570  D RECNO564
571  D RECNO565
572  D RECNO566
573  D RECNO567
574  D RECNO568
575  D RECNO569
576  D RECNO570
577  D RECNO571
578  D RECNO572
579  D RECNO573
580  D RECNO574
581  D RECNO575
582  D RECNO576
583  D RECNO577
584  D RECNO578
585  D RECNO579
586  D RECNO580
587  D RECNO581
588  D RECNO582
589  D RECNO583
590  D RECNO584
591  D RECNO585
592  D RECNO586
593  D RECNO587
594  D RECNO588
595  D RECNO589
596  D RECNO590
597  D RECNO591
598  D RECNO592
599  D RECNO593
510  D RECNO594
511  D RECNO595
512  D RECNO596
513  D RECNO597
514  D RECNO598
515  D RECNO599
516  D RECNO600
517  D RECNO601
518  D RECNO602
519  D RECNO603
520  D RECNO604
521  D RECNO605
522  D RECNO606
523  D RECNO607
524  D RECNO608
525  D RECNO609
526  D RECNO610
527  D RECNO611
528  D RECNO612
529  D RECNO613
530  D RECNO614
531  D RECNO615
532  D RECNO616
533  D RECNO617
534  D RECNO618
535  D RECNO619
536  D RECNO620
537  D RECNO621
538  D RECNO622
539  D RECNO623
540  D RECNO624
541  D RECNO625
542  D RECNO626
543  D RECNO627
544  D RECNO628
545  D RECNO629
546  D RECNO630
547  D RECNO631
548  D RECNO632
549  D RECNO633
550  D RECNO634
551  D RECNO635
552  D RECNO636
553  D RECNO637
554  D RECNO638
555  D RECNO639
556  D RECNO640
557  D RECNO641
558  D RECNO642
559  D RECNO643
560  D RECNO644
561  D RECNO645
562  D RECNO646
563  D RECNO647
564  D RECNO648
565  D RECNO649
566  D RECNO650
567  D RECNO651
568  D RECNO652
569  D RECNO653
570  D RECNO654
571  D RECNO655
572  D RECNO656
573  D RECNO657
574  D RECNO658
575  D RECNO659
576  D RECNO660
577  D RECNO661
578  D RECNO662
579  D RECNO663
580  D RECNO664
581  D RECNO665
582  D RECNO666
583  D RECNO667
584  D RECNO668
585  D RECNO669
586  D RECNO670
587  D RECNO671
588  D RECNO672
589  D RECNO673
590  D RECNO674
591  D RECNO675
592  D RECNO676
593  D RECNO677
594  D RECNO678
595  D RECNO679
596  D RECNO680
597  D RECNO681
598  D RECNO682
599  D RECNO683
510  D RECNO684
511  D RECNO685
512  D RECNO686
513  D RECNO687
514  D RECNO688
515  D RECNO689
516  D RECNO690
517  D RECNO691
518  D RECNO692
519  D RECNO693
520  D RECNO694
521  D RECNO695
522  D RECNO696
523  D RECNO697
524  D RECNO698
525  D RECNO699
526  D RECNO700
527  D RECNO701
528  D RECNO702
529  D RECNO703
530  D RECNO704
531  D RECNO705
532  D RECNO706
533  D RECNO707
534  D RECNO708
535  D RECNO709
536  D RECNO710
537  D RECNO711
538  D RECNO712
539  D RECNO713
540  D RECNO714
541  D RECNO715
542  D RECNO716
543  D RECNO717
544  D RECNO718
545  D RECNO719
546  D RECNO720
547  D RECNO721
548  D RECNO722
549  D RECNO723
550  D RECNO724
551  D RECNO725
552  D RECNO726
553  D RECNO727
554  D RECNO728
555  D RECNO729
556  D RECNO730
557  D RECNO731
558  D RECNO732
559  D RECNO733
560  D RECNO734
561  D RECNO735
562  D RECNO736
563  D RECNO737
564  D RECNO738
565  D RECNO739
566  D RECNO740
567  D RECNO741
568  D RECNO742
569  D RECNO743
570  D RECNO744
571  D RECNO745
572  D RECNO746
573  D RECNO747
574  D RECNO748
575  D RECNO749
576  D RECNO750
577  D RECNO751
578  D RECNO752
579  D RECNO753
580  D RECNO754
581  D RECNO755
582  D RECNO756
583  D RECNO757
584  D RECNO758
585  D RECNO759
586  D RECNO760
587  D RECNO761
588  D RECNO762
589  D RECNO763
590  D RECNO764
591  D RECNO765
592  D RECNO766
593  D RECNO767
594  D RECNO768
595  D RECNO769
596  D RECNO770
597  D RECNO771
598  D RECNO772
599  D RECNO773
510  D RECNO774
511  D RECNO775
512  D RECNO776
513  D RECNO777
514  D RECNO778
515  D RECNO779
516  D RECNO780
517  D RECNO781
518  D RECNO782
519  D RECNO783
520  D RECNO784
521  D RECNO785
522  D RECNO786
523  D RECNO787
524  D RECNO788
525  D RECNO789
526  D RECNO790
527  D RECNO791
528  D RECNO792
529  D RECNO793
530  D RECNO794
531  D RECNO795
532  D RECNO796
533  D RECNO797
534  D RECNO798
535  D RECNO799
536  D RECNO800
537  D RECNO801
538  D RECNO802
539  D RECNO803
540  D RECNO804
541  D RECNO805
542  D RECNO806
543  D RECNO807
544  D RECNO808
545  D RECNO809
546  D RECNO810
547  D RECNO811
548  D RECNO812
549  D RECNO813
550  D RECNO814
551  D RECNO815
552  D RECNO816
553  D RECNO817
554  D RECNO818
555  D RECNO819
556  D RECNO820
557  D RECNO821
558  D RECNO822
559  D RECNO823
560  D RECNO824
561  D RECNO825
562  D RECNO826
563  D RECNO827
564  D RECNO828
565  D RECNO829
566  D RECNO830
567  D RECNO831
568  D RECNO832
569  D RECNO833
570  D RECNO834
571  D RECNO835
572  D RECNO836
573  D RECNO837
574  D RECNO838
575  D RECNO839
576  D RECNO840
577  D RECNO841
578  D RECNO842
579  D RECNO843
580  D RECNO844
581  D RECNO845
582  D RECNO846
583  D RECNO847
584  D RECNO848
585  D RECNO849
586  D RECNO850
587  D RECNO851
588  D RECNO852
589  D RECNO853
590  D RECNO854
591  D RECNO855
592  D RECNO856
593  D RECNO857
594  D RECNO858
595  D RECNO859
596  D RECNO860
597  D RECNO861
598  D RECNO862
599  D RECNO863
510  D RECNO864
511  D RECNO865
512  D RECNO866
513  D RECNO867
514  D RECNO868
515  D RECNO869
516  D RECNO870
517  D RECNO871
518  D RECNO872
519  D RECNO873
520  D RECNO874
521  D RECNO875
522  D RECNO876
523  D RECNO877
524  D RECNO878
525  D RECNO879
526  D RECNO880
527  D RECNO881
528  D RECNO882
529  D RECNO883
530  D RECNO884
531  D RECNO885
532  D RECNO886
533  D RECNO887
534  D RECNO888
535  D RECNO889
536  D RECNO890
537  D RECNO891
538  D RECNO892
539  D RECNO893
540  D RECNO894
541  D RECNO895
542  D RECNO896
543  D RECNO897
544  D RECNO898
545  D RECNO899
546  D RECNO900
547  D RECNO901
548  D RECNO902
549  D RECNO903
550  D RECNO904
551  D RECNO905
552  D RECNO906
553  D RECNO907
554  D RECNO908
555  D RECNO909
556  D RECNO910
557  D RECNO911
558  D RECNO912
559  D RECNO913
560  D RECNO914
561  D RECNO915
562  D RECNO916
563  D RECNO917
564  D RECNO918
565  D RECNO919
566  D RECNO920
567  D RECNO921
568  D RECNO922
569  D RECNO923
570  D RECNO924
571  D RECNO925
572  D RECNO926
573  D RECNO927
574  D RECNO928
575  D RECNO929
576  D RECNO930
577  D RECNO931
578  D RECNO932
579  D RECNO933
580  D RECNO934
581  D RECNO935
582  D RECNO936
583  D RECNO937
584  D RECNO938
585  D RECNO939
586  D RECNO940
587  D RECNO941
588  D RECNO942
589  D RECNO943
590  D RECNO944
591  D RECNO945
592  D RECNO946
593  D RECNO947
594  D RECNO948
595  D RECNO949
596  D RECNO950
597  D RECNO951
598  D RECNO952
599  D RECNO953
510  D RECNO954
511  D RECNO955
512  D RECNO956
513  D RECNO957
514  D RECNO958
515  D RECNO959
516  D RECNO960
517  D RECNO961
518  D RECNO962
519  D RECNO963
520  D RECNO964
521  D RECNO965
522  D RECNO966
523  D RECNO967
524  D RECNO968
525  D RECNO969
526  D RECNO970
527  D RECNO971
528  D RECNO972
529  D RECNO973
530  D RECNO974
531  D RECNO975
532  D RECNO976
533  D RECNO977
534  D RECNO978
535  D RECNO979
536  D RECNO980
537  D RECNO981
538  D RECNO982
539  D RECNO983
540  D RECNO984
541  D RECNO985
542  D RECNO986
543  D RECNO987
544  D RECNO988
545  D RECNO989
546  D RECNO990
547  D RECNO991
548  D RECNO992
549  D RECNO993
550  D RECNO994
551  D RECNO995
552  D RECNO996
553  D RECNO997
554  D RECNO998
555  D RECNO999
556  D RECNO1000
557  D RECNO1001
558  D RECNO1002
559  D RECNO1003
560  D RECNO1004
561  D RECNO1005
562  D RECNO1006
563  D RECNO1007
564  D RECNO1008
565  D RECNO1009
566  D RECNO1010
567  D RECNO1011
568  D RECNO1012
569  D RECNO1013
570  D RECNO1014
571  D RECNO1015
572  D RECNO1016
573  D RECNO1017
574  D RECNO1018
575  D RECNO1019
576  D RECNO1020
577  D RECNO1021
578  D RECNO1022
579  D RECNO1023
580  D RECNO1024
581  D RECNO1025
582  D RECNO1026
583  D RECNO1027
584  D RECNO1028
585  D RECNO1029
586  D RECNO1030
587  D RECNO1031
588  D RECNO1032
589  D RECNO1033
590  D RECNO1034
591  D RECNO1035
592  D RECNO1036
593  D RECNO1037
594  D RECNO1038
595  D RECNO1039
596  D RECNO1040
597  D RECNO1041
598  D RECNO1042
599  D RECNO1043
510  D RECNO1044
511  D RECNO1045
512  D RECNO1046
513  D RECNO1047
514  D RECNO1048
515  D RECNO1049
516  D RECNO1050
517  D RECNO1051
518  D RECNO1052
519  D RECNO1053
520  D RECNO1054
521  D RECNO1055
522  D RECNO1056
523  D RECNO1057
524  D RECNO1058
525  D RECNO1059
526  D RECNO1060
527  D RECNO1061
528  D RECNO1062
529  D RECNO1063
530  D RECNO1064
531  D RECNO1065
532  D RECNO1066
533  D RECNO1067
534  D RECNO1068
535  D RECNO1069
536  D RECNO1070
537  D RECNO1071
538  D RECNO1072
539  D RECNO1073
540  D RECNO1074
541  D RECNO1075
542  D RECNO1076
543  D RECNO1077
544  D RECNO1078
545  D RECNO1079
546  D RECNO1080
547  D RECNO1081
548  D RECNO1082
549  D RECNO1083
550  D RECNO1084
551  D RECNO1085
552  D RECNO1086
553  D RECNO1087
554  D RECNO1088
555  D RECNO1089
556  D RECNO1090
557  D RECNO1091
558  D RECNO1092
559  D RECNO1093

```

## COVF

```

1   ▶ R←SIZE COVF F,OIO;K,L,L,K,R1
2   ▶ FUNCTION TO GENERATE TERMS IN 2D COVARIANCE FCN FOR IMAGE
3   ▶
4   ▶ F←F-MEAN F
5   ▶ OIO←0
6   ▶ L←SIZE[0]
7   ▶ K←SIZE[1]
8   ▶ R←R1←SIZE[0]
9   ▶ I←0
10  ▶ LOOP1: K←0
11  ▶ LOOP2: R[L,K]←(+/+/(+L,K)↑F)×((-(L,K))↑F))-x/pF
12  ▶ R1[L,K]←(+/+/(+L,-K)↑F)×((-(L,-K))↑F))-x/pF
13  ▶ →(K>K+1)/LOOP2
14  ▶ →(C>L+1)/LOOP1
15  ▶ R←7@ 0 1 →R1),R

```

## CORR

```

1   ▶ RI←SIZE CORRF F,OIO,R,L;K,L,R
2   ▶ FUNCTION TO ESTIMATE 2D COVARIANCE FUNCTION FOR IMAGE
3   ▶
4   ▶ OIO←0
5   ▶ ←SIZE(OI,F)
6   ▶ ←SIZE(R)
7   ▶ ←SIZE,L;K,L,R
8   ▶ I←0
9   ▶ LOOP1: K←0
10  ▶ LOOP2: R[L,K]←(-K)↑(-K)ΦR[L,]
11  ▶ →(K>K+1)/LOOP2
12  ▶ →(L>L+1)/LOOP1
13

```

## MVLEV

```

1   ▶ A←MVLEV RPT,RT,N,NB,NDIM,AA1,AF,AB:AF1,AB1,EF,EB,GAMF,GAMB,DELT,I,OIO
2   ▶ RPT IS THE (MATRIX) CORRELATION FUNCTION. PLANE J CONTAINS R(j,j).
3   ▶ TO OBTAIN FORWARD PARAMETERS: A←MVLEV [0], BACKWARD AB←MVLEV [1].
4   ▶ ERROR COVARIANCE EF←MVLEV [2]. FU←MVLEV [3] TO OBTAIN INNOVATIONS.
5   ▶ FILTER: LIKewise FOR BACKWARD PARAMETERS.
6   ▶
7   ▶ OIO←0
8   ▶ →((+RPT)[0])^3=+RPT)/START
9   ▶ 'INVALID ARGUMENT'
10  ▶ →0
11  ▶
12  ▶ START:N←+((+RPT)[0])
13  ▶ NB←(+RPT)[1]
14  ▶ RT←0.2^4 RPT
15  ▶ ((NB))←1/NB
16  ▶ +(N)1//RECURS
17  ▶ AF1←(1,NB,NB)↑I
18  ▶ AB1←2*I
19  ▶ RPT←RPT[0,1]
20  ▶
21  ▶ REGHALG
22  ▶ REFHJ←AA1←MVLEV(N,NB)+RPT
23  ▶ AA1←AA1↑0,1
24  ▶ AB1←AB1↑0,1
25  ▶ AF1←AF1↑0,1
26  ▶ DELT←DELT↑0,1
27  ▶ AB1←AB1↑1,2
28  ▶ AA1←AA1↑1,2
29  ▶ DELT←DELT↑1,2
30  ▶ DELT←((NDIM,NB)↑0,1 0 0 ×RPT)↑1,2×(NDIM,NB)↑0,AF
31  ▶ GAMF←(EEF)↑1,2×DELT
32  ▶ GAMB←(EEF)↑1,2×DELT
33  ▶ AF←(AF1,[0,0])-e(AF1,[0,0])↑1,2×GAMF
34  ▶ AB←(AB1,[0,0])-(e(AB1,[0,0])↑1,2×GAMB
35  ▶ NDIM←NEXN+1
36  ▶ AF1←((NDIM,NB)↑0,1)+,X(NDIM,NB)↑0,AF
37  ▶ AB1←((NDIM,NB)↑0,1)+,X(NDIM,NB)↑0,AB
38  ▶ BA←AF,[0,5] AB

```

## FF2DLEV

```

[1] # AFFECTS ONLY N, AA, AF, L, R, R1, R2, R3, R4
[2]
[3] # FUNCTION TO DO 2D LEVTRSM RECURRENCE = IMAGE FUNCTION ANALOG
[4] # CIO-->
[5] AA+MVLEV R
[6] AF+AF[0,0,0]
[7] EN+AF[0,0,0]
[8] M=(2K+1)
[9] AF[0,0,0,1M]=IM
[10] S=M=1,0
[11] A0=(BEN[+,XG]
[12] AF(AFT,XA0)=AO[0]
[13] AO[0,0]=AO[0]
[14]
[15] #COVF[0,0,0]
[16] # R+SIZE COVF F;CIO,K,L,L;R1
[17] # FUNCTION TO GENERATE TERMS IN 2D COVARIANCE FCN FOR IMAGE
[18]
[19] #F=F-MEAN F
[20] #CIO=0
[21] #L=SIZE[0]
[22] #R=SIZE[1]
[23] #F=R*FSIZE*0
[24] #L=0
[25] LOOP1 K=0
[26] LOOP2 R(L,K)=+(+/r/((L-K)+F)*X((-L-K)+F))-X/-F
[27] R1(L,K)=+(+/-((L,-K)+F)/X((-L,-K)+F))-X/-F
[28] +(F)+F+1)/LOOP2
[29] +(F)L-L+1)/LOOP1
[30] R+(F) 0 1 +R1),R

```

## APPENDIX B

### DERIVATION OF THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE TWO POLE AUTOREGRESSIVE MODEL

#### Power Spectrum

$$|H(e^{j\omega})|^2 = H(e^{j\omega}) \cdot (H(e^{j\omega}))^* = H(e^{j\omega})H(e^{-j\omega})$$

In this case:

$$H(e^{j\omega}) = \frac{1}{1+\alpha e^{-j\omega}} \quad H(e^{-j\omega}) = \frac{1}{1+\alpha e^{j\omega}}$$

Calculating  $H(e^{j\omega})H(e^{-j\omega})$ :

$$\frac{1}{(1+\alpha e^{-j\omega})(1+\alpha e^{j\omega})} = \frac{1}{1+\alpha e^{-j\omega} + \alpha e^{j\omega} + \alpha^2} = \frac{1}{1+2\alpha \cos(\omega) + \alpha^2}$$

The final result is:

$$S_Y(\omega) = |H(e^{j\omega})|^2 = \frac{1}{(1+\alpha^2)+2\alpha \cos(\omega)} \quad (B.1)$$

#### Autocorrelation Function

Starting with  $H(z)$  for this case:

$$H(z) = \frac{1}{1+\alpha z^{-1}}$$

Per Ref. 7:p. 158:

$$z^{-1}[H(z)] = h(n) = (-\alpha)^n \cdot u(n) \quad \text{for } \alpha < 1 \\ (\text{u}(n) \text{ is the unit step function})$$

Per Ref. 5:pp. 391-395, for the white noise input case:

$$R_Y(\ell) = \sum_{n=-\infty}^{\infty} h(n) \cdot h(n-\ell) \quad (B.2)$$

Substituting  $h(n)$  above into Eq. B.2

$$R_Y(\ell) = \sum_{n=\ell}^{\infty} (-\alpha)^n \cdot (-\alpha)^{n-\ell} \quad \ell \geq 0$$

$$= \frac{1}{(-\alpha)^\ell} \sum_{n=\ell}^{\infty} (-\alpha)^{2n} \quad \ell \geq 0$$

The summation term may also be expressed as:

$$\sum_{n=\ell}^{\infty} (-\alpha)^{2n} = \sum_{n=0}^{\infty} (-\alpha)^{2n} - \sum_{n=0}^{\ell-1} (-\alpha)^{2n}$$

Per Ref. 12:p. 8, the summation terms on the right are equal to:

$$\sum_{n=0}^{\infty} (-\alpha)^{2n} = \frac{1}{1 - (-\alpha)^2} \quad (\alpha < 1) \quad \sum_{n=0}^{\ell-1} (-\alpha)^{2n} = \frac{1 - (-\alpha)^{2\ell}}{1 - (-\alpha)^2} \quad (\alpha < 1)$$

As a result:

$$\sum_{n=\ell}^{\infty} (-\alpha)^{2n} = \frac{1}{1 - (-\alpha)^2} - \frac{1 - (-\alpha)^{2\ell}}{1 - (-\alpha)^2} = \frac{(-\alpha)^{2\ell}}{1 - (-\alpha)^2} \quad (\alpha < 1)$$

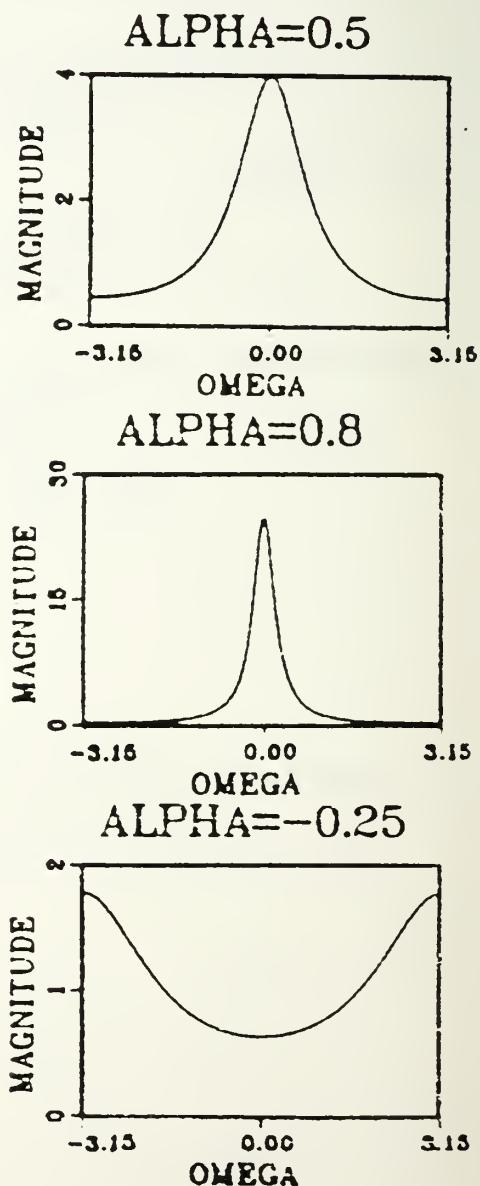
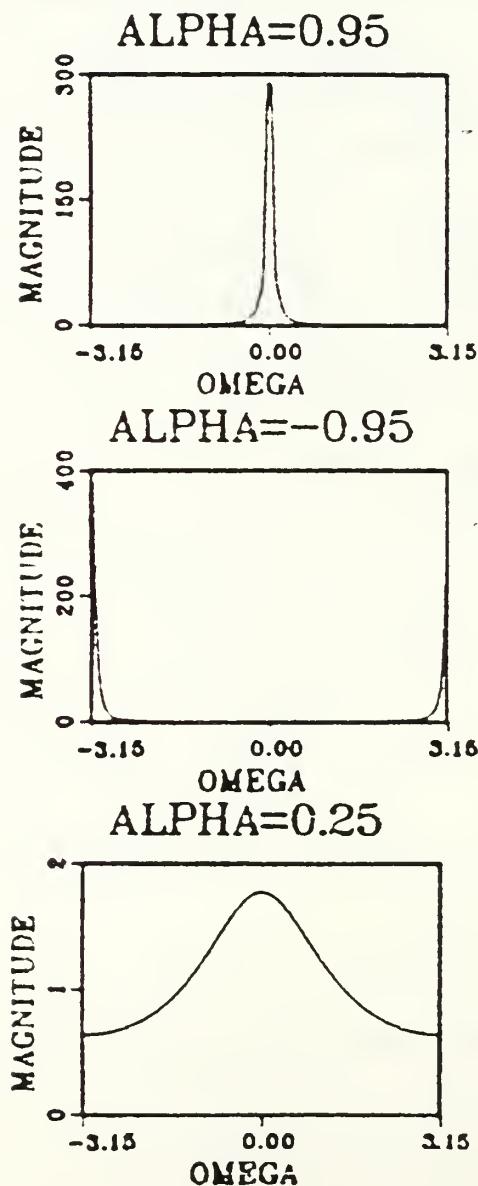
Substituting and using  $(-\alpha)^2 = \alpha^2$  yields:

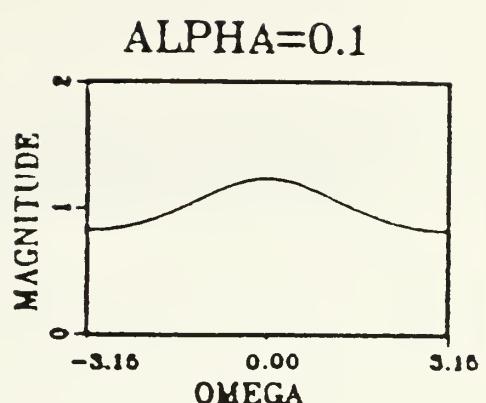
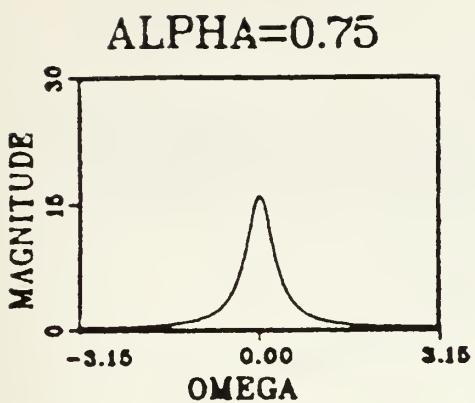
$$R_Y(\ell) = \frac{1}{(-\alpha)^\ell} \cdot \frac{(-\alpha)^{2\ell}}{1 - \alpha^2} = \frac{(-\alpha)^\ell}{1 - \alpha^2} \quad \ell \geq 0 \text{ and } \alpha < 1 \quad (\text{B.3})$$

## APPENDIX C

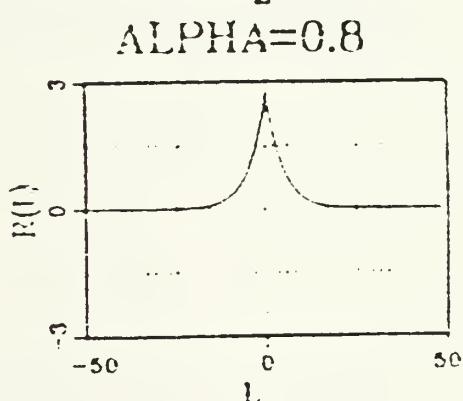
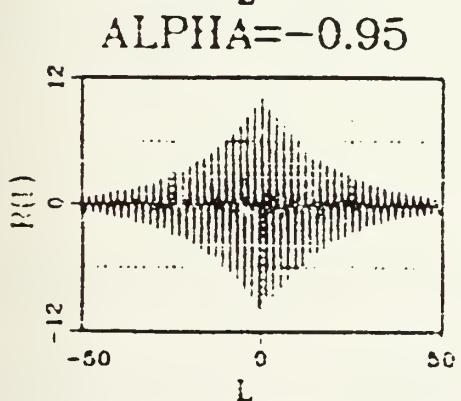
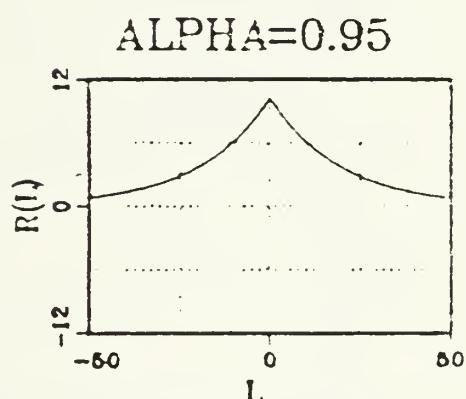
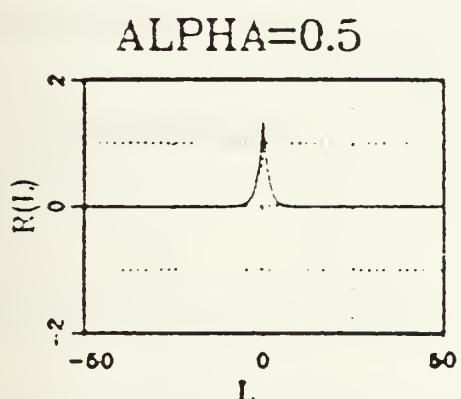
### GRAPHICAL RESULTS FOR THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE TWO POLE AUTOREGRESSIVE MODEL

Power Spectrum

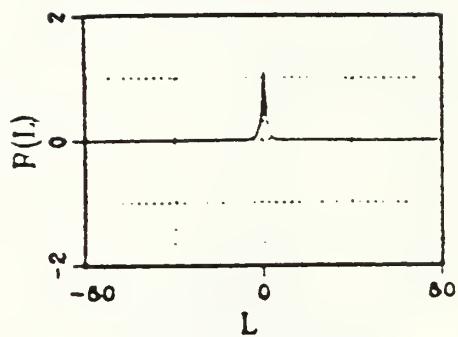




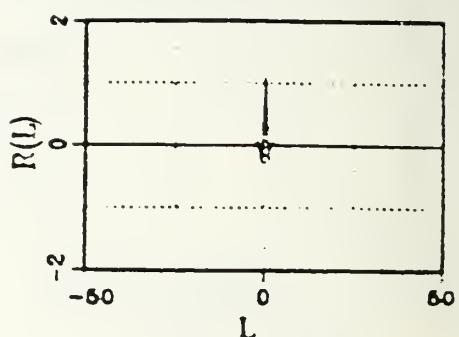
Autocorrelation Function



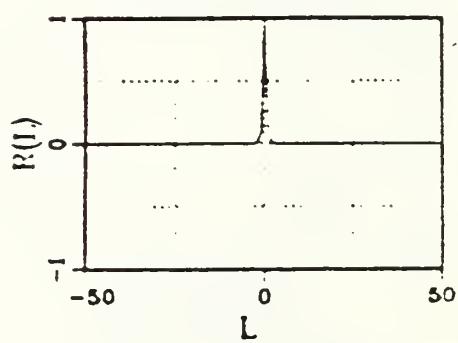
$\text{ALPHA}=0.25$



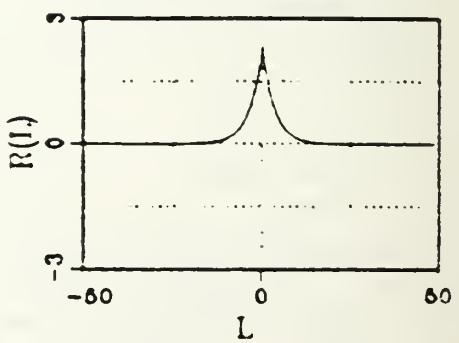
$\text{ALPHA}=-0.25$



$\text{ALPHA}=0.1$



$\text{ALPHA}=0.75$



## APPENDIX D

### DERIVATION OF THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE FOUR POLE AUTOREGRESSIVE MODEL

#### Power Spectrum

$$|H(e^{j\omega})|^2 = H(e^{j\omega}) \cdot H(e^{-j\omega})$$

In this case:

$$\begin{aligned} H(e^{-j\omega}) &= \frac{1}{(1-\alpha e^{-j\theta} e^{j\omega})(1-\alpha e^{j\theta} e^{j\omega})} = \frac{1}{1-\alpha e^{j(\theta+\omega)} - \alpha e^{-j(\theta-\omega)} + \alpha^2 e^{j2\omega}} \\ H(e^{j\omega}) &= \frac{1}{(1-\alpha e^{-j\theta} e^{-j\omega})(1-\alpha e^{j\theta} e^{-j\omega})} = \frac{1}{1-\alpha e^{-j(\theta+\omega)} - \alpha e^{j(\theta-\omega)} + \alpha^2 e^{-j2\omega}} \end{aligned}$$

Multiplying the above expressions yields:

$$\begin{aligned} H(e^{j\omega}) H(e^{-j\omega}) &= \frac{1}{1-\alpha e^{j(\theta-\omega)} - \alpha e^{-j(\theta+\omega)} + \alpha^2 e^{-j2\omega} - \alpha e^{j(\theta+\omega)} + \alpha^2 e^{j2\theta} + \alpha^2} \\ &\quad - \alpha^3 e^{j(\theta-\omega)} - \alpha e^{-j(\theta-\omega)} + \alpha^2 e^{-j2\theta} - \alpha^3 e^{-j(\theta-\omega)} \\ &\quad + \alpha^2 e^{j2\omega} - \alpha^3 e^{j(\theta+\omega)} - \alpha^3 e^{-j(\theta+\omega)} + \alpha^4 \end{aligned}$$

Combining terms:

$$\begin{aligned} H(e^{j\omega}) \cdot H(e^{-j\omega}) &= \frac{1}{1+2\alpha^2 + \alpha^4 - \alpha^3 (e^{-j(\theta-\omega)} + e^{j(\theta-\omega)}) + \alpha^2 (e^{j2\theta} + e^{-j2\theta})} \\ &\quad - \alpha^3 (e^{j(\theta+\omega)} + e^{-j(\theta+\omega)}) + \alpha^2 (e^{j2\omega} + e^{-j2\omega}) \\ &\quad - \alpha (e^{j(\theta-\omega)} + e^{-j(\theta-\omega)}) - \alpha (e^{j(\theta+\omega)} + e^{-j(\theta+\omega)}) \end{aligned}$$

Using Euler's relation and combining terms:

$$\begin{aligned} H(e^{j\omega}) \cdot H(e^{-j\omega}) &= \frac{1}{1+\alpha^4 + 2(\alpha^2 - \cos(\theta-\omega)) [\alpha^3 + \alpha] + \alpha^2 \cos(2\theta) - \cos(\theta+\omega)} \\ &\quad \times [\alpha^3 + \alpha] + \alpha^2 \cos(2\omega) \end{aligned}$$

Using  $\cos(\theta - \omega) + \cos(\theta + \omega) = 2\cos(\theta)\cos(\omega)$ , and since  $S_Y(\omega) = H(e^{j\omega}) \cdot H(e^{-j\omega}) \cdot \sigma^2$ , with  $\sigma^2 = 1$  the final result is:

$$S_Y(\omega) = \frac{1}{1+\alpha^4+2(\alpha^2-2[\alpha^3-\alpha]\cos(\theta)\cos(\omega)+\alpha^2(\cos(2\theta)+\cos(2\omega)))} \quad (D.1)$$

### Autocorrelation Function

$$H(z) = \frac{1}{1-\alpha e^{j\theta}z^{-1}-\alpha e^{-j\theta}z^{-1}+\alpha^2 z^{-2}} = \frac{1}{1-(e^{j\theta}+e^{-j\theta})z^{-1}+\alpha^2 z^{-2}}$$

Using Euler's relation:

$$H(z) = \frac{1}{1-2\alpha\cos(\theta)z^{-1}+\alpha^2 z^{-2}} \quad |z| > |\alpha|$$

Per Ref. 7:pp. 204-216, partial fraction expansion can be used to find the inverse Z transform. To do so  $H(z)$  can be expressed in the form:

$$H(z) = \frac{1}{(1-\alpha e^{-j\theta}z^{-1})(1-\alpha e^{j\theta}z^{-1})} = \frac{z}{(z-\alpha e^{-j\theta})(z-\alpha e^{j\theta})}$$

Using the partial fraction expansion and table look up [Ref. 7:p. 158] yields:

$$h(n) = \frac{\alpha^n}{\sin(\theta)} \cos(n\theta + \theta - \frac{\pi}{2}) \cdot u(n) \quad \text{for } \alpha < 1 \quad (D.2)$$

Since  $\cos(\theta - \frac{\pi}{2}) = \sin(\theta)$ , the final expression for  $h(n)$  is:

$$h(n) = \frac{\alpha^n}{\sin(\theta)} \sin((n+1)\theta) \cdot u(n) \quad \alpha < 1 \quad (D.3)$$

For simplicity in further derivation of  $R_y(\ell)$  based on  $h(n)$ , Eq. D.2 will be used.

Using the expression for the autocorrelation function of a random process represented by the above filter with a white noise input [Ref. 4:pp. 391-395]:

$$R_y(\ell) = \sum_{n=-\infty}^{\infty} h(n) \cdot h(n-\ell) = \frac{\alpha^{-\ell}}{\sin^2(\theta)} \sum_{n=\ell}^{\infty} \alpha^{2n} \cdot \cos(n\theta + \theta - \frac{\pi}{2}) \cdot \cos((n-\ell)\theta + \theta - \frac{\pi}{2}) \quad (D.4)$$

$n = \ell$  in the summation index since  $h(n)$  is causal.  $\ell$  is assumed to be greater than zero here. For  $\ell < 0$ ,  $R_y(\ell) = R_y(-\ell)$  by symmetry of the autocorrelation function [Ref. 5:p. 388], so we can proceed assuming only positive values of  $\ell$ .

Using the trigonometric identity for a product of cosines:

$$R_y(\ell) = \frac{\alpha^{-\ell}}{\sin^2(\theta)} \sum_{n=\ell}^{\infty} \alpha^{2n} [\frac{1}{2} \cdot \cos(\ell\theta) + \frac{1}{2} \cos(2n\theta - \ell\theta + 2\theta - \pi)] \quad \ell \geq 0$$

$$R_y(\ell) = \frac{\cos(\ell\theta) \cdot \alpha^{-\ell}}{2\sin^2(\theta)} \cdot \sum_{n=\ell}^{\infty} \alpha^{2n} + \frac{\alpha^{-\ell}}{2\sin^2(\theta)} \sum_{n=\ell}^{\infty} \alpha^{2n} \cdot (2n\theta - \ell\theta + 2\theta - \pi) \quad (D.5)$$

$\ell \geq 0$

Using  $\sum_{n=\ell}^{\infty} \rightarrow \sum_{n=0}^{\infty} - \sum_{n=0}^{\ell-1}$  and standard geometric progression identities [Ref. 12:p. 8]:

$$\sum_{n=0}^{\infty} \alpha^n = \frac{1}{1-\alpha} \quad \text{and} \quad \sum_{n=0}^{\ell-1} \alpha^n = \frac{1-\alpha^\ell}{1-\alpha}$$

For the first term in  $R_Y(\ell)$

$$\begin{aligned} \sum_{n=\ell}^{\infty} \alpha^{2n} &= \sum_{n=0}^{\infty} \alpha^{2n} - \sum_{n=0}^{\ell-1} \alpha^{2n} = \frac{1}{1-\alpha^2} - \frac{1-\alpha^{2\ell}}{1-\alpha^2} \\ &= \frac{\alpha^{2\ell}}{1-\alpha^2} \end{aligned} \quad (\text{D.6})$$

For the second term in  $R_Y(\ell)$ :

$$\text{let } \phi = -\ell\theta + 2\theta - \pi$$

Using Euler's relation:

$$\begin{aligned} \cos(2n\theta + \phi) &= \frac{e^{j(2n\theta+\phi)} + e^{-j(2n\theta+\phi)}}{2} \\ \sum_{n=\ell}^{\infty} \alpha^{2n} \cdot \cos(2n\theta + \phi) &= \frac{1}{2} \sum_{n=\ell}^{\infty} \alpha^{2n} [e^{j(2n\theta+\phi)} + e^{-j(2n\theta+\phi)}] \\ &= \frac{1}{2} \sum_{n=\ell}^{\infty} \alpha^{2n} \cdot e^{j(2n\theta+\phi)} + \frac{1}{2} \sum_{n=\ell}^{\infty} \alpha^{2n} \cdot e^{-j(2n\theta+\phi)} \end{aligned} \quad (\text{D.7})$$

For large  $n$ , it is evident that the  $\alpha^{2n}$  term will tend to make the term in each sum approach 0 for  $\alpha < 1$ , and thus ensures convergence and a closed form expression for each term. Pursuing the mathematics required to find this closed form expression we have:

$$\frac{1}{2} \sum_{n=\ell}^{\infty} \alpha^{2n} \cdot e^{j(2n\theta+\phi)} = \frac{e^{j\phi}}{2} \sum_{n=\ell}^{\infty} (\alpha e^{j\theta})^{2n} = \frac{e^{j\phi}}{2} \left[ \sum_{n=0}^{\infty} (\alpha e^{j\theta})^{2n} - \sum_{n=0}^{\ell-1} (\alpha e^{j\theta})^{2n} \right]$$

$$\begin{aligned}
 \frac{1}{2} \sum_{n=\ell}^{\infty} \alpha^{2n} \cdot e^{j(2n\theta+\phi)} &= \frac{e^{j\phi}}{2} \left[ \frac{1}{1-(\alpha e^{j\theta})^2} - \frac{1-(\alpha e^{j\theta})^{2\ell}}{1-(\alpha e^{j\theta})^2} \right] \\
 &= \frac{\alpha^{2\ell} \cdot e^{j(2\theta\ell+\phi)}}{2(1-(\alpha e^{j\theta})^2)}
 \end{aligned} \tag{D.8}$$

For the conjugate term we must have:

$$\frac{1}{2} \sum_{n=\ell}^{\infty} \alpha^{2n} \cdot e^{-j(2n\theta+\phi)} = \frac{\alpha^{2\ell} \cdot e^{-j(2\theta\ell+\phi)}}{2(1-(\alpha e^{-j\theta})^2)} \tag{D.9}$$

Next a common denominator must be found to sum these two terms:

$$\begin{aligned}
 &\frac{\alpha^{2\ell} \cdot e^{j(2\theta\ell+\phi)}}{2(1-(\alpha e^{j\theta})^2)} \cdot \frac{(1-(\alpha e^{-j\theta})^2)}{(1-(\alpha e^{-j\theta})^2)} \\
 &= \frac{\alpha^{2\ell} e^{j(2\theta\ell+\phi)} - \alpha^{(2\ell+2)} e^{j(2\theta\ell-2\theta+\phi)}}{2(1-(\alpha e^{j\theta})^2 - (\alpha e^{-j\theta})^2 + \alpha^4)} \\
 \\ 
 &\frac{\alpha^{2\ell} \cdot e^{-j(2\theta\ell+\phi)}}{2(1-(\alpha e^{-j\theta})^2)} \cdot \frac{(1-(\alpha e^{j\theta})^2)}{(1-(\alpha e^{j\theta})^2)} \\
 &= \frac{\alpha^{2\ell} e^{-j(2\theta\ell+\phi)} - \alpha^{(2\ell+2)} e^{-j(2\theta\ell-2\theta+\phi)}}{2(1-(\alpha e^{j\theta})^2 - (\alpha e^{-j\theta})^2 + \alpha^4)}
 \end{aligned}$$

By Euler's relation:

$$1 - (\alpha e^{j\theta})^2 - (\alpha e^{-j\theta})^2 + \alpha^4 = 1 - 2\alpha^2 \cos(2\theta) + \alpha^4$$

Adding the terms with the common denominator yields:

$$\frac{\alpha^{2\ell} [e^{j(2\theta\ell+\phi)} - \alpha^2 e^{j(2\theta\ell-2\theta+\phi)} + e^{-j(2\theta\ell+\phi)} - \alpha^2 e^{-j(2\theta\ell-2\theta+\phi)}]}{2(1 - 2\alpha^2 \cos(2\theta) + \alpha^4)}$$

Again using Euler's relation the above expression reduces to:

$$\frac{\alpha^{2l} [\cos(2\theta l + \phi) - \alpha^2 \cos(2\theta l - 2\theta + \phi)]}{1 - 2\alpha^2 \cos(2\theta) + \alpha^4} \quad (\text{D.10})$$

which is the sum of the last two terms in Eq. (D.7). Substituting Eq. (D.10) and Eq. (D.6) into Eq. (D.5) yields:

$$\begin{aligned} R_Y^{(l)} &= \frac{\cos(l\theta) \cdot \alpha^{-l}}{2\sin^2(\theta)} \left[ \frac{\alpha^{2l}}{1-\alpha^2} \right] \\ &\quad + \frac{\alpha^{-l}}{2\sin^2(\theta)} \left[ \frac{\alpha^{2l} [\cos(2\theta l + \phi) - \alpha^2 \cos(2\theta l - 2\theta + \phi)]}{1 - 2\alpha^2 \cos(2\theta) + \alpha^4} \right] \end{aligned} \quad (\text{D.11})$$

Combining and canceling terms and substituting for  $\phi$  and noticing that the same result must hold for  $l < 0$  we have:

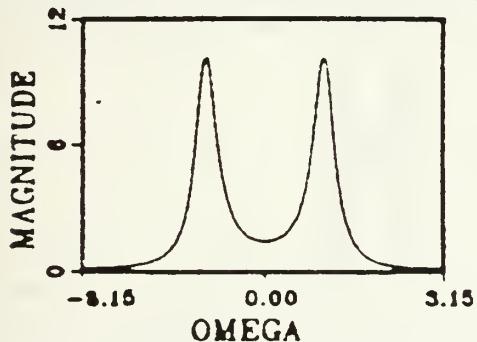
$$R_Y^{(l)} = \frac{\alpha^l}{2\sin^2(\theta)} \left[ \frac{\cos(|l|\theta)}{1-\alpha^2} + \frac{\cos((2+|l|)\theta - \pi) - \alpha^2 \cos(|l|\theta - \pi)}{1 - 2\alpha^2 \cos(2\theta) + \alpha^4} \right] \quad (\text{D.12})$$

## APPENDIX E

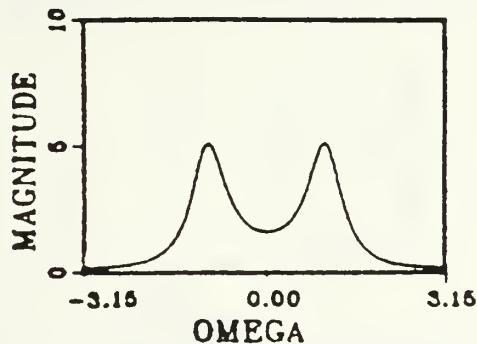
### GRAPHICAL RESULTS FOR THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE FOUR POLE AUTOREGRESSIVE MODEL

Power Spectrum

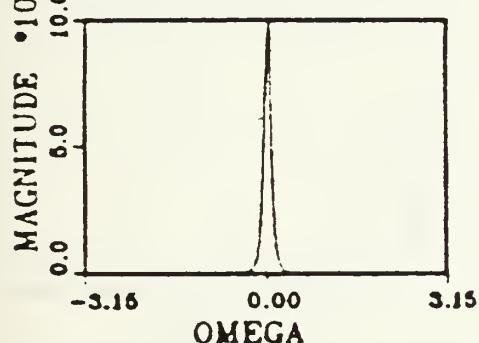
**ALPHA=0.8 THETA=PI/3**



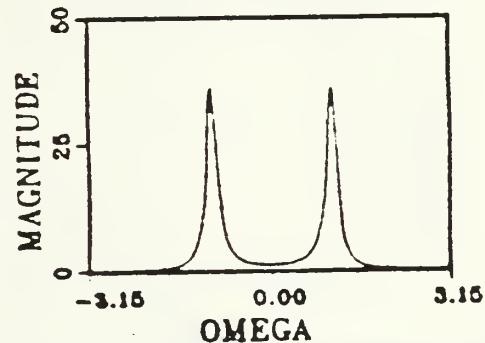
**ALPHA=0.7 THETA=PI/3**



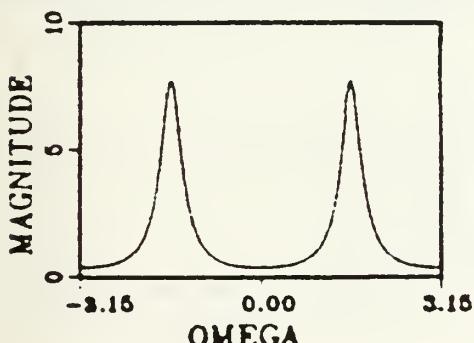
**ALPHA=0.9 THETA=0**



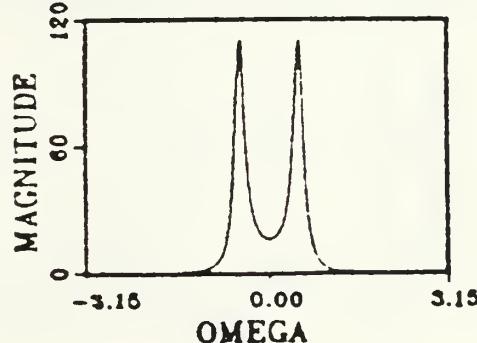
**ALPHA=0.9 THETA=PI/3**



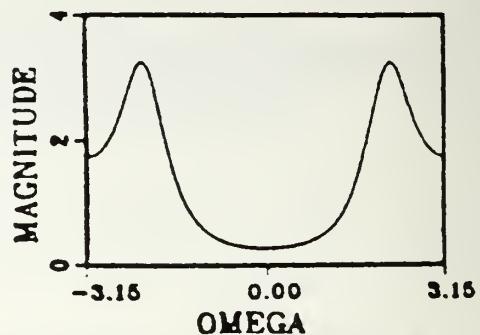
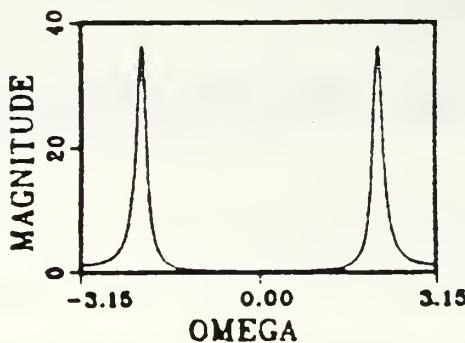
**ALPHA=0.8 THETA=PI/2**



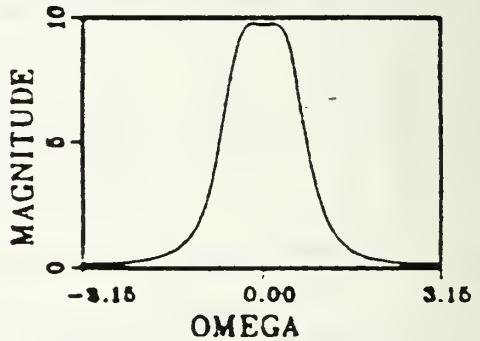
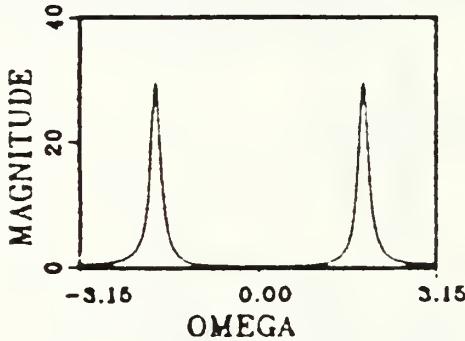
**ALPHA=0.9 THETA=PI/6**



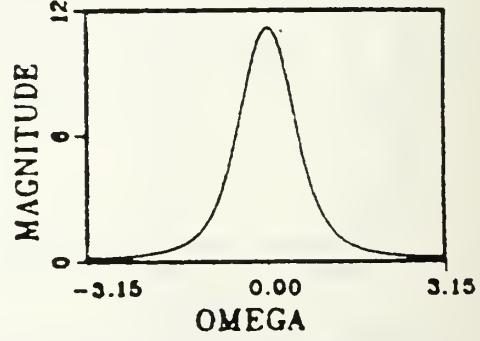
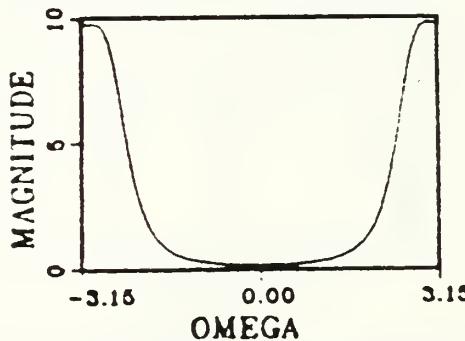
ALPHA=0.9 THETA=2\*PI/3 ALPHA=0.6 THETA=2\*PI/3



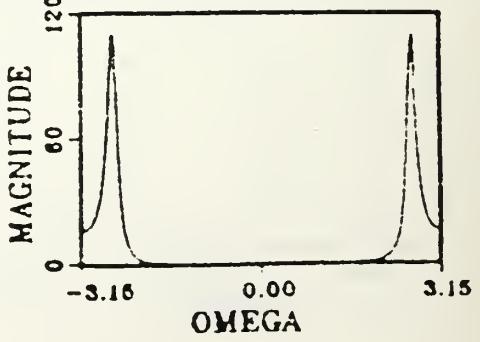
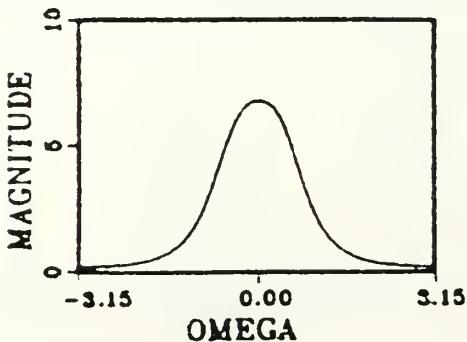
ALPHA=0.9 THETA=7\*PI/12 ALPHA=0.6 THETA=PI/6



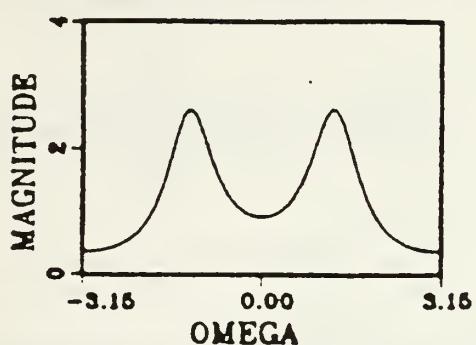
ALPHA=0.6 THETA=5\*PI/6 ALPHA=0.5 THETA=PI/10



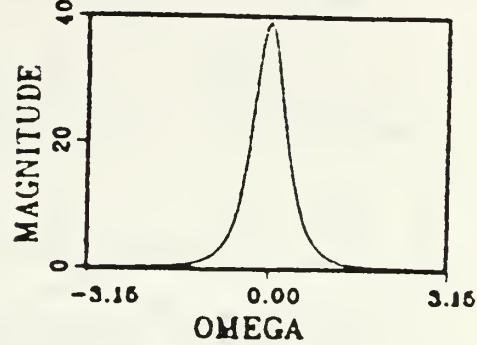
ALPHA=0.5 THETA=PI/6 ALPHA=0.9 THETA=5\*PI/6



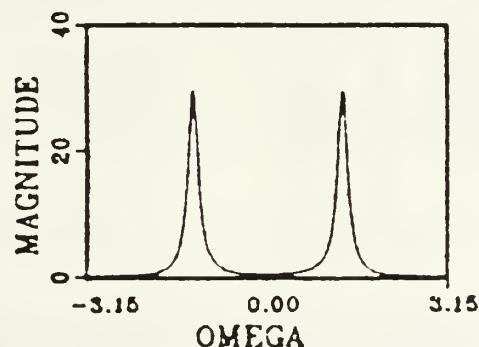
ALPHA=0.6 THETA=5\*PI/12



ALPHA=0.6 THETA=0

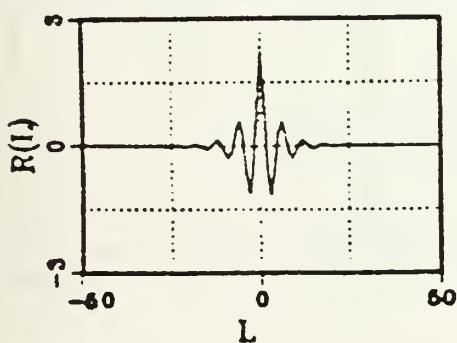


ALPHA=0.9 THETA=5\*PI/12

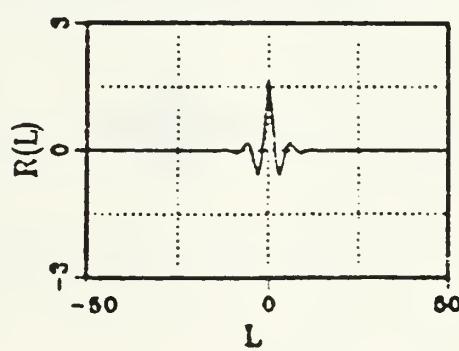


### Autocorrelation Function

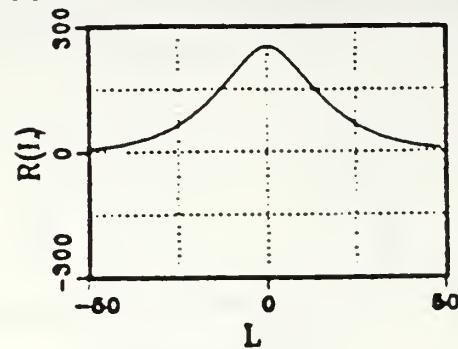
ALPHA=0.8 THETA=PI/3



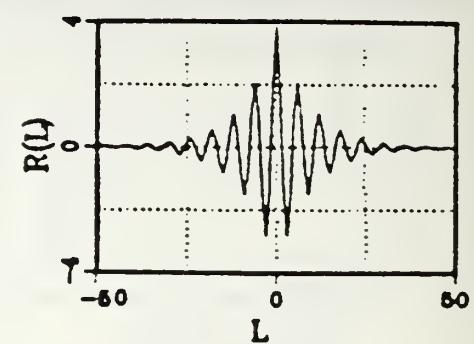
ALPHA=0.7 THETA=PI/3



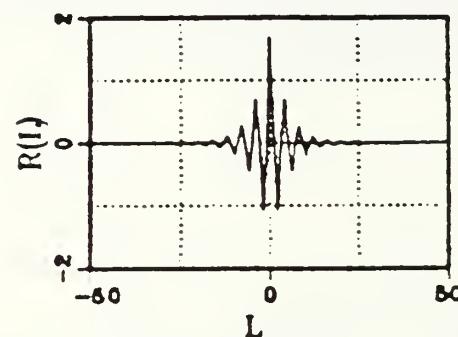
ALPHA=0.9 THETA=0



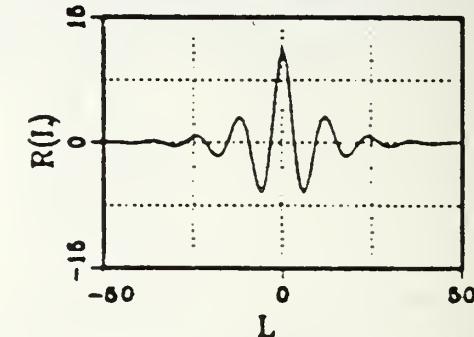
ALPHA=0.9 THETA=PI/3



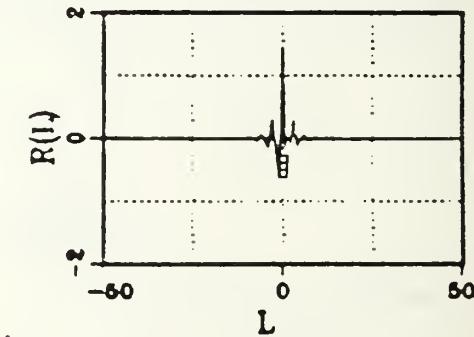
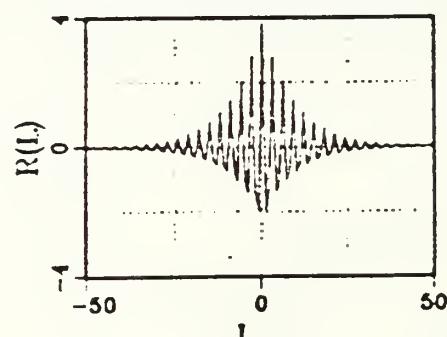
ALPHA=0.8 THETA=PI/2



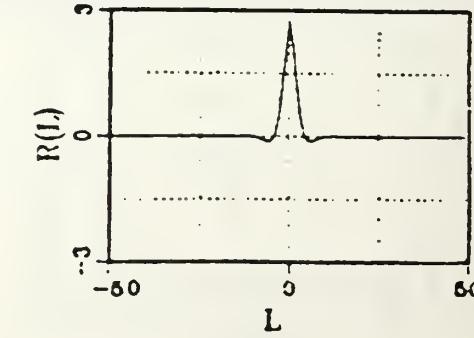
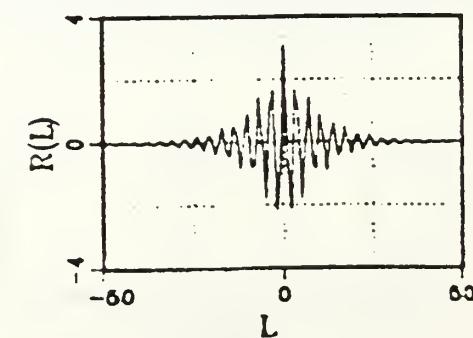
ALPHA=0.9 THETA=PI/6



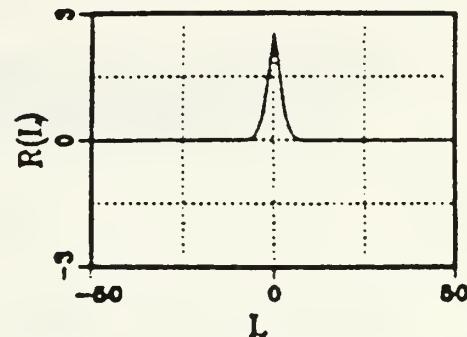
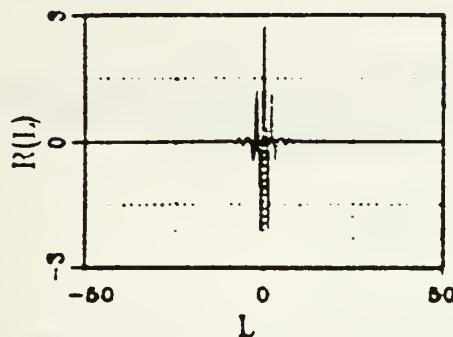
ALPHA=0.9 THETA=2\*PI/3 ALPHA=0.6 THETA=2\*PI/3



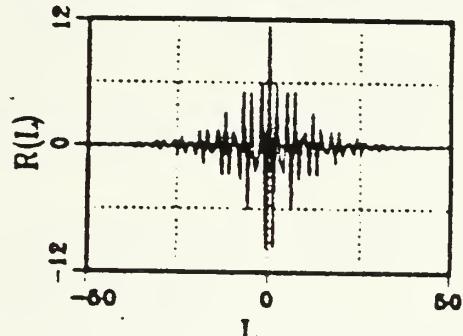
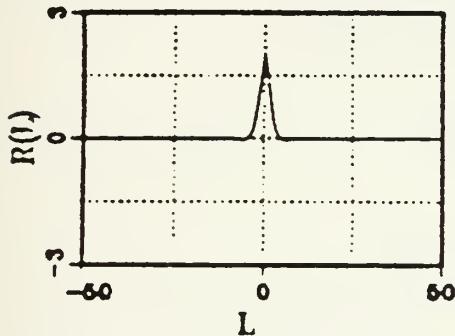
ALPHA=0.9 THETA=7\*PI/12 ALPHA=0.6 THETA=PI/6



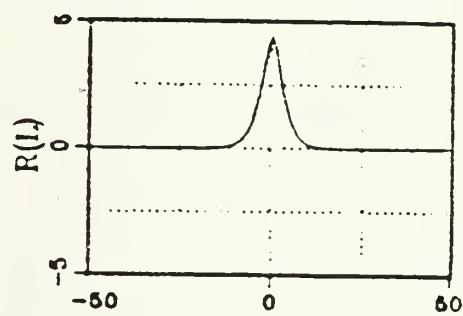
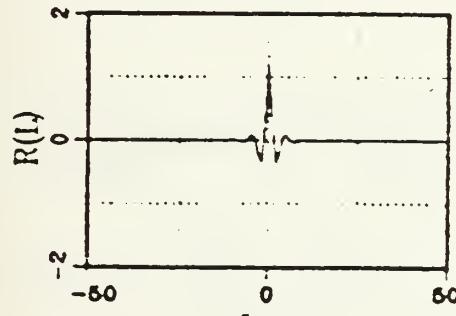
ALPHA=0.6 THETA=5\*PI/6 ALPHA=0.5 THETA=PI/10



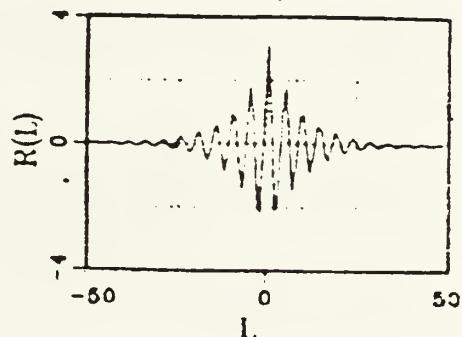
ALPHA=0.5 THETA=PI/6 ALPHA=0.9 THETA=5\*PI/6



ALPHA=0.6 THETA=5\*PI/12 ALPHA=0.6 THETA=0



ALPHA=0.9 THETA=5\*PI/12



## APPENDIX F

### DERIVATION OF THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE FOUR POLE AUTOREGRESSIVE MODEL (WITH TWO POLES ON THE REAL AXIS)

#### Power Spectrum

Using initial results from Appendix D with the necessary modifications (including  $\theta = 0$ ) we have:

$$H(e^{-j\omega}) = \frac{1}{1 - \alpha_a e^{j\omega} - \alpha_b e^{j\omega} + \alpha_a \alpha_b e^{j2\omega}}$$

$$H(e^{j\omega}) = \frac{1}{1 - \alpha_a e^{-j\omega} - \alpha_b e^{-j\omega} + \alpha_a \alpha_b e^{-j2\omega}}$$

Therefore:

$$\begin{aligned} H(e^{j\omega}) \cdot H(e^{-j\omega}) &= \frac{1}{1 - \alpha_a e^{-j\omega} - \alpha_b e^{-j\omega} + \alpha_a \alpha_b e^{-j2\omega} - \alpha_a^2 e^{j\omega} + \alpha_a^2 \alpha_b^2 + \alpha_a^2 \alpha_b^2} \\ &\quad - \alpha_a^2 \alpha_b e^{-j\omega} - \alpha_b e^{j\omega} + \alpha_a^2 \alpha_b^2 + \alpha_b^2 - \alpha_a^2 \alpha_b^2 e^{-j\omega} \\ &\quad + \alpha_a^2 \alpha_b e^{j2\omega} - \alpha_a^2 \alpha_b^2 e^{j\omega} - \alpha_a^2 \alpha_b^2 e^{j\omega} + \alpha_a^2 \alpha_b^2 \end{aligned}$$

Combining terms and using Euler's relation:

$$\begin{aligned} H(e^{j\omega}) \cdot H(e^{-j\omega}) &= \frac{1}{1 - (\alpha_a + \alpha_b + \alpha_a^2 \alpha_b^2 + \alpha_a^2 \alpha_b^2) \cdot 2 \cos(\omega) + 2 \alpha_a \alpha_b \cos(2\omega) + \alpha_a^2} \\ &\quad + 2 \alpha_a \alpha_b + \alpha_b^2 + \alpha_a^2 \alpha_b^2 \end{aligned}$$

Assuming  $\sigma^2 = 1$  and since  $S_Y(\omega) = H(e^{j\omega}) \cdot H(e^{-j\omega}) \cdot \sigma^2$ , the final result is:

$$S_Y(\omega) = \frac{1}{1 - 2(\alpha_a + \alpha_b + \alpha_a^2 \alpha_b + \alpha_a \alpha_b^2) \cos(\omega) + 2\alpha_a \alpha_b \cos(2\omega) + \alpha_a^2 + 2\alpha_a \alpha_b + \alpha_b^2} \quad (F.1)$$

### Autocorrelation Function

$$H(z) = \frac{1}{1 - \alpha_a e^{j\theta} z^{-1} - \alpha_b e^{-j\theta} z^{-1} + \alpha_a \alpha_b z^{-2}}$$

Letting  $\theta = 0$

$$H(z) = \frac{1}{1 - (\alpha_a + \alpha_b)z^{-1} + \alpha_a \alpha_b z^{-2}} = \frac{z^2}{z^2 - (\alpha_a + \alpha_b)z + \alpha_a \alpha_b}$$

$$|z| > |\alpha|$$

Expanding in terms of partial fractions we have:

$$H(z) = \frac{\alpha_a}{\alpha_a - \alpha_b} \cdot \frac{z}{z - \alpha_a} + \frac{\alpha_b}{\alpha_b - \alpha_a} \cdot \frac{z}{z - \alpha_b}$$

This corresponds to the impulse response

$$h(n) = \left( \frac{\alpha_a}{\alpha_a - \alpha_b} \cdot \alpha_a^n + \frac{\alpha_b}{\alpha_b - \alpha_a} \cdot \alpha_b^n \right) \cdot u(n) \quad \alpha < 1 \quad (F.2)$$

Proceeding as in Appendix D:

$$R_Y(\ell) = \sum_{n=-\infty}^{\infty} h(n) \cdot h(n-\ell) = \sum_{n=\ell}^{\infty} \left( \frac{\alpha_a^{n+1}}{\alpha_a - \alpha_b} + \frac{\alpha_b^{n+1}}{\alpha_b - \alpha_a} \right) \left( \frac{\alpha_a^{n+1-\ell}}{\alpha_a - \alpha_b} + \frac{\alpha_b^{n+1-\ell}}{\alpha_b - \alpha_a} \right)$$

$(\ell \geq 0)$

$$R_Y(\ell) = \sum_{n=\ell}^{\infty} \left[ \frac{\alpha_a^{2n+2-\ell}}{(\alpha_a - \alpha_b)^2} + \frac{\alpha_a^{n+1} \cdot \alpha_b^{n+1-\ell}}{(\alpha_a - \alpha_b)(\alpha_b - \alpha_a)} + \frac{\alpha_a^{n+1-\ell} \cdot \alpha_b^{n+1}}{(\alpha_b - \alpha_a)(\alpha_a - \alpha_b)} \right. \\ \left. + \frac{\alpha_b^{2n+2-\ell}}{(\alpha_b - \alpha_a)^2} \right] \quad (\ell \geq 0) \quad (F.3)$$

Since  $(\alpha_a - \alpha_b) = -(\alpha_b - \alpha_a)$ :

$$(\alpha_a - \alpha_b)^2 = (\alpha_b - \alpha_a)^2 = -(\alpha_a - \alpha_b)(\alpha_b - \alpha_a) = -(\alpha_b - \alpha_a)(\alpha_a - \alpha_b)$$

So:

$$R_Y(\ell) = \frac{1}{(\alpha_a - \alpha_b)^2} \left[ \alpha_a^{2-\ell} \sum_{n=\ell}^{\infty} \alpha_a^{2n} - \alpha_a \alpha_b^{1-\ell} \sum_{n=\ell}^{\infty} (\alpha_a \alpha_b)^n \right. \\ \left. - \alpha_b \alpha_a^{1-\ell} \sum_{n=\ell}^{\infty} (\alpha_b \alpha_a)^n + \alpha_b^{2-\ell} \sum_{n=\ell}^{\infty} \alpha_b^{2n} \right] \quad (\ell \geq 0)$$

Continuing with the same principles and assumptions as in Appendix D, we have:

$$\sum_{n=\ell}^{\infty} \alpha^{2n} = \sum_{n=0}^{\infty} \alpha^{2n} - \sum_{n=0}^{\ell-1} \alpha^{2n} = \frac{1}{1-\alpha^2} - \frac{1-\alpha^{2\ell}}{1-\alpha^2} = \frac{\alpha^{2\ell}}{1-\alpha^2}$$

$$\sum_{n=\ell}^{\infty} \alpha^n = \sum_{n=0}^{\infty} \alpha^n - \sum_{n=0}^{\ell-1} \alpha^n = \frac{1}{1-\alpha} - \frac{1-\alpha^\ell}{1-\alpha} = \frac{\alpha^\ell}{1-\alpha}$$

Making the appropriate substitutions in the expression for  $R_Y(\ell)$ , we have:

$$R_Y(\ell) = \frac{1}{(\alpha_a - \alpha_b)^2} [\alpha_a^{2-\ell} \left( \frac{\alpha_a^{2\ell}}{1-\alpha_a^2} \right) - \alpha_a^{\ell-1} \left( \frac{\alpha_a^\ell \alpha_b^\ell}{1-\alpha_a \alpha_b} \right) \\ - \alpha_b^{\ell-1} \left( \frac{\alpha_a^\ell \alpha_b^\ell}{1-\alpha_a \alpha_b} \right) + \alpha_b^{2-\ell} \left( \frac{\alpha_b^{2\ell}}{1-\alpha_b^2} \right)] \quad (\ell \geq 0) \quad (F.4)$$

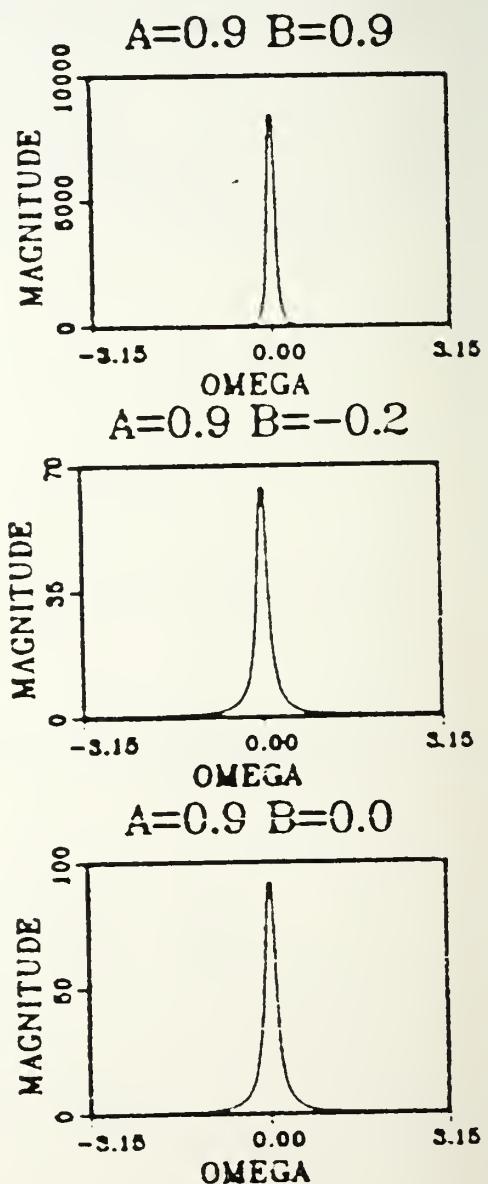
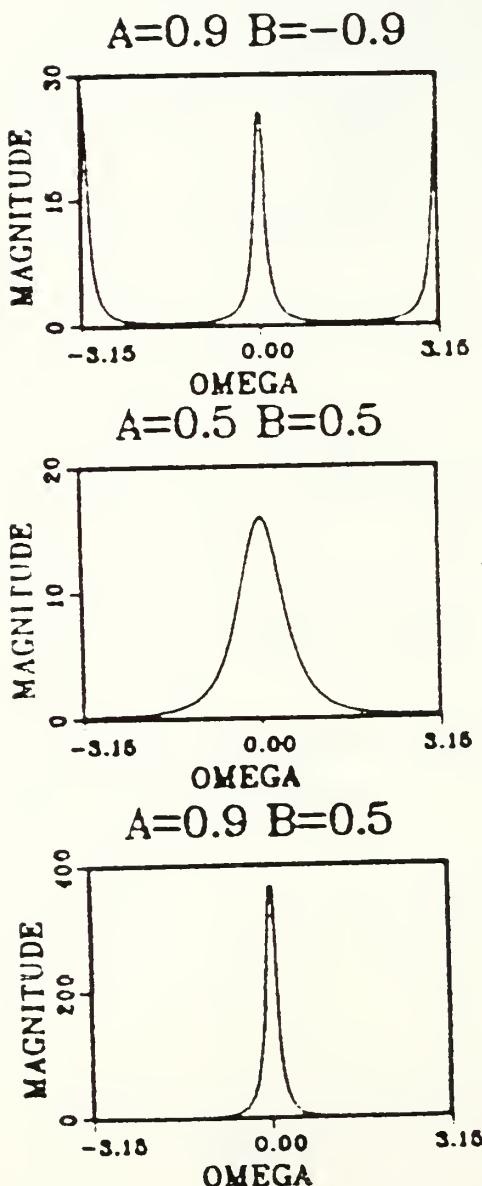
Combining terms yields the final expression:

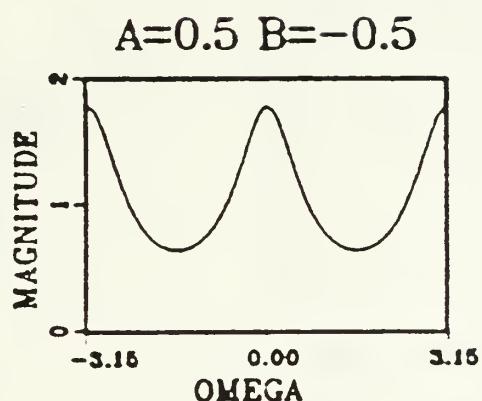
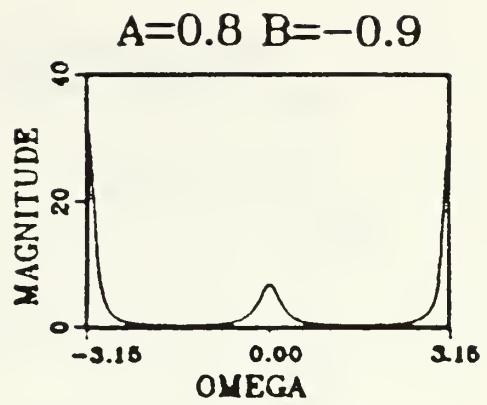
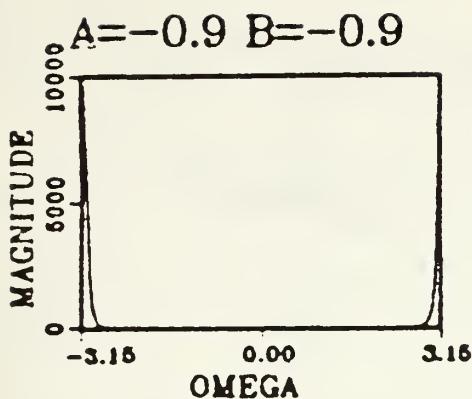
$$R_Y(\ell) = \frac{1}{(\alpha_a - \alpha_b)^2} \left[ \frac{\alpha_a^{2+\ell}}{1-\alpha_a^2} - \frac{\alpha_a^{\ell+1} \alpha_b + \alpha_a \alpha_b^{\ell+1}}{1-\alpha_a \alpha_b} + \frac{\alpha_b^{2+\ell}}{1-\alpha_b^2} \right] \quad (F.5)$$

APPENDIX G

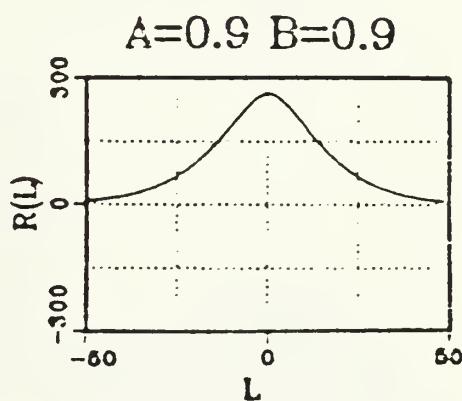
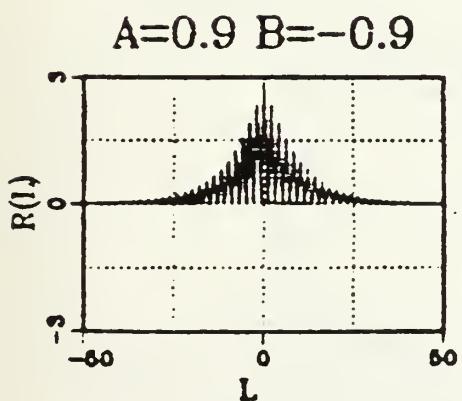
GRAPHICAL RESULTS FOR THE POWER SPECTRUM AND AUTO-CORRELATION FUNCTION FOR THE FOUR POLE AUTO-REGRESSIVE MODEL (WITH TWO POLES ON THE REAL AXIS)

Power Spectrum

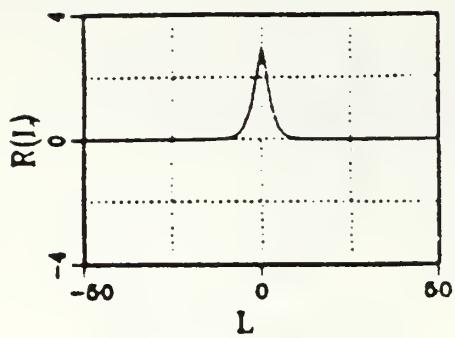




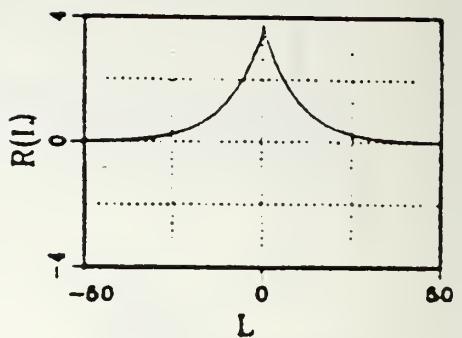
Autocorrelation Function



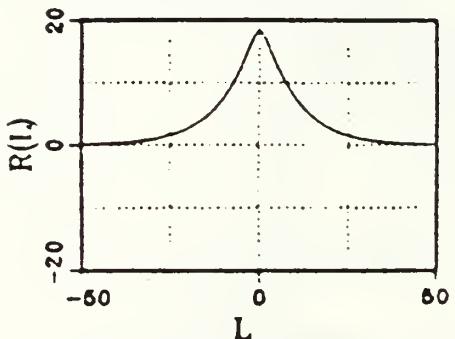
$A=0.5$   $B=0.5$



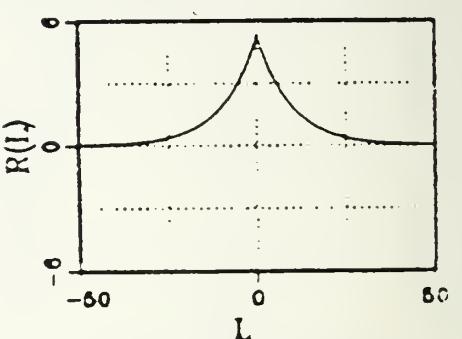
$A=0.9$   $B=-0.2$



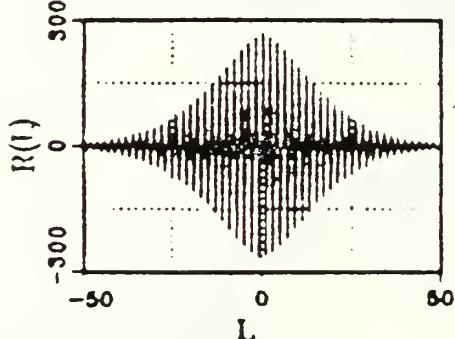
$A=0.9$   $B=0.5$



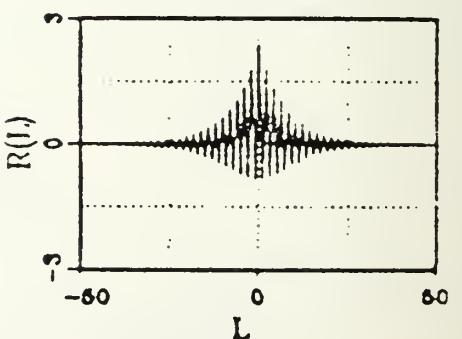
$A=0.9$   $B=0.0$



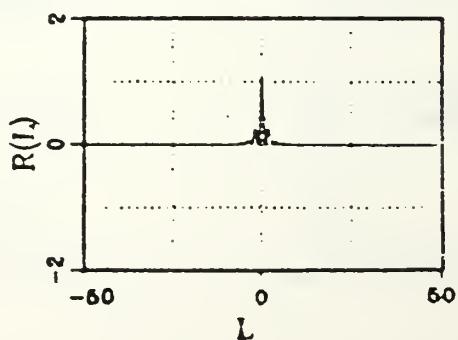
$A=-0.9$   $B=-0.9$



$A=0.8$   $B=-0.9$

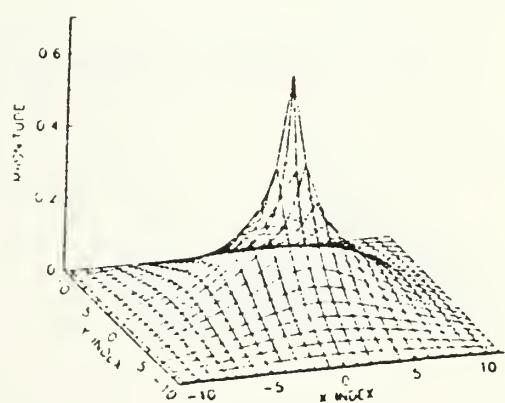
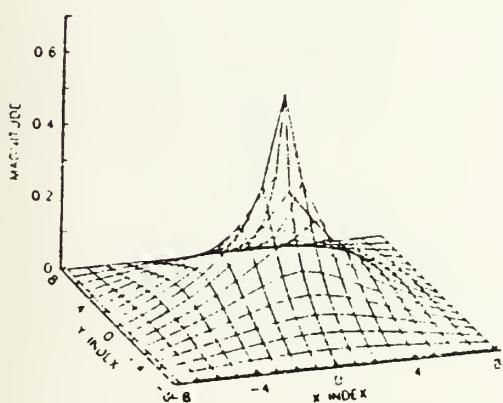
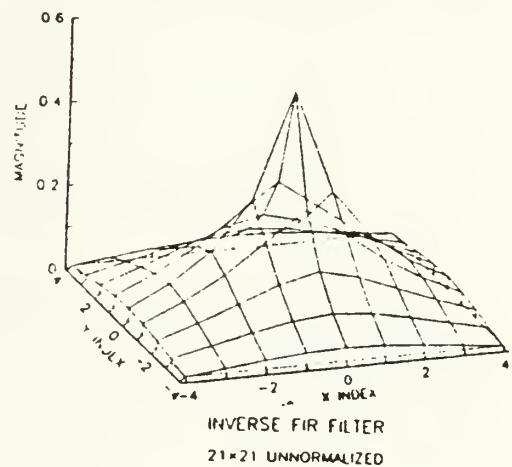
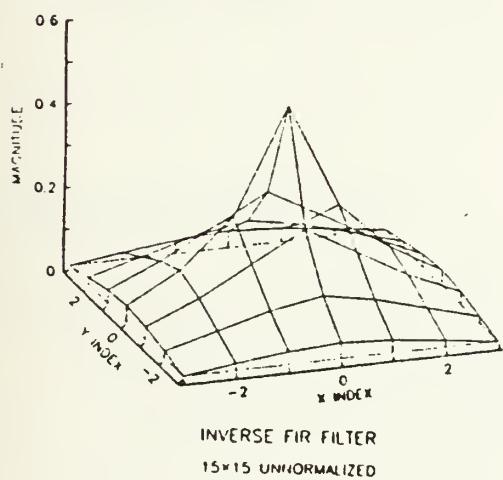
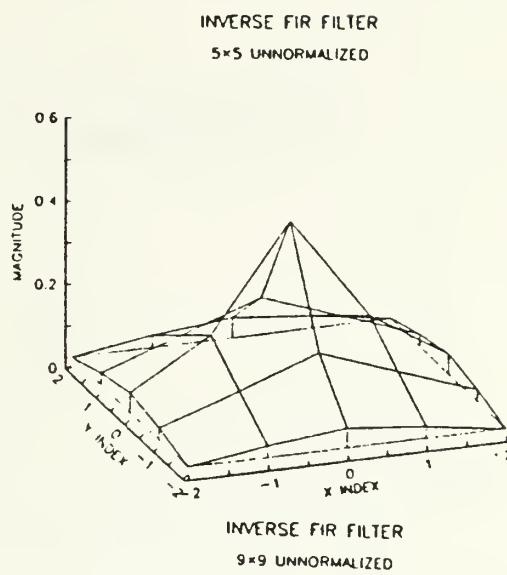
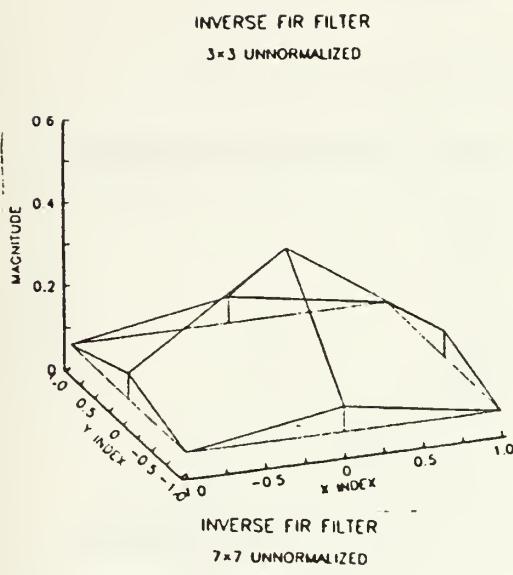


$A=0.5$   $B=-0.5$

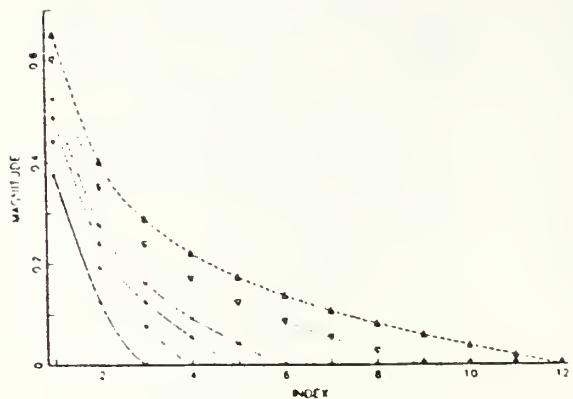


## APPENDIX H

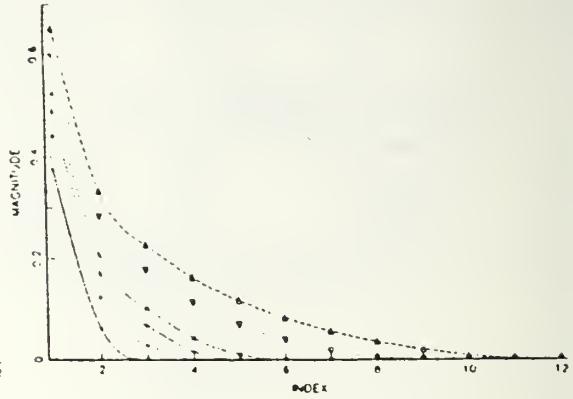
### LAPLACIAN INVERSE FILTER FORMS



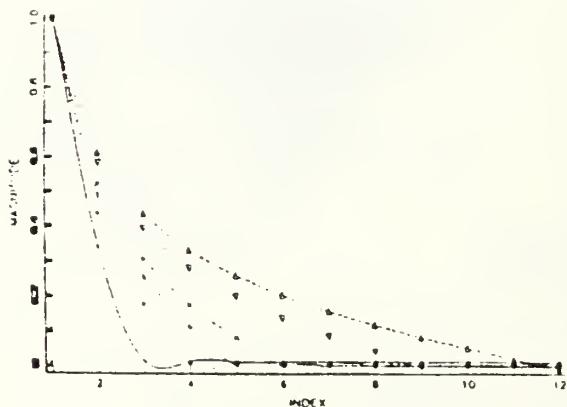
LAPLACIAN INVERSE FILTERS (3×3 TO 21×21)  
HORIZONTAL CROSS SECTION



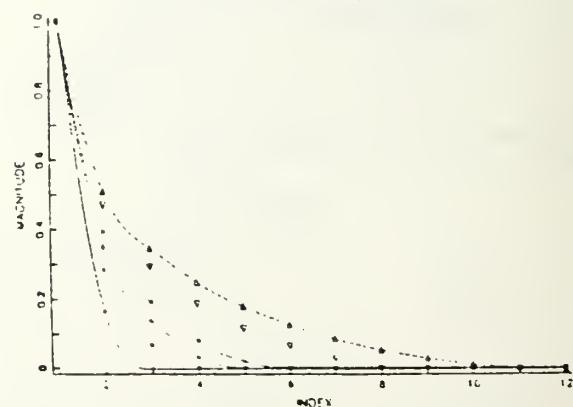
LAPLACIAN INVERSE FILTERS (3×3 TO 21×21)  
DIAGONAL CROSS SECTION



LAPLACIAN INVERSE FILTERS (3×3 TO 21×21)  
HORIZONTAL CROSS SECTION (NORMALIZED)



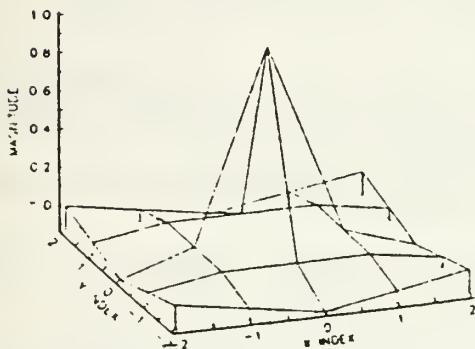
LAPLACIAN INVERSE FILTERS (3×3 TO 21×21)  
DIAGONAL CROSS SECTION (NORMALIZED)



## APPENDIX I

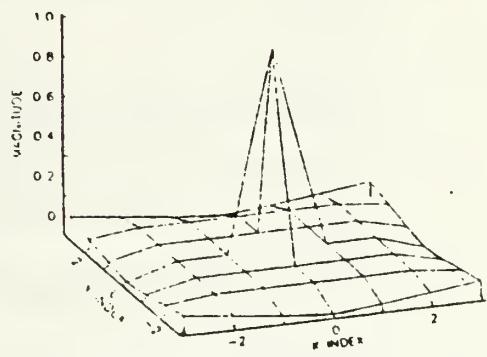
### CONVOLUTION OF LAPLACIAN DIFFERENCE OPERATOR AND VARIOUS SIZE FIR INVERSE FILTERS

CONVOLUTION OF LAPLACIAN AND ITS  $3 \times 3$  INVERSE

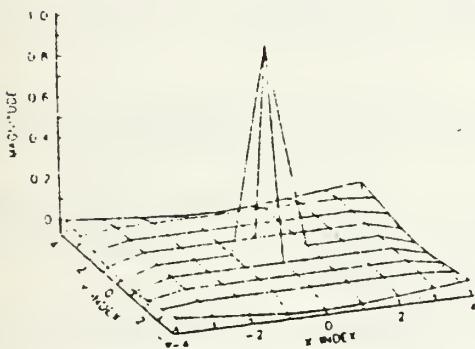


CONVOLUTION OF LAPLACIAN AND ITS  $7 \times 7$  INVERSE

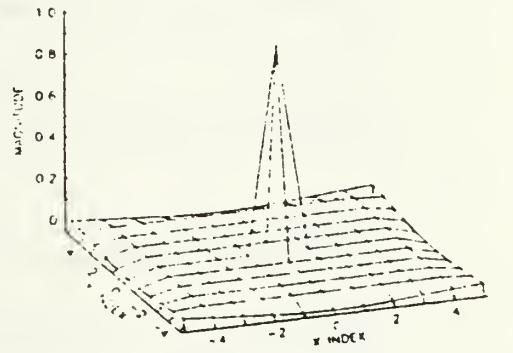
CONVOLUTION OF LAPLACIAN AND ITS  $5 \times 5$  INVERSE



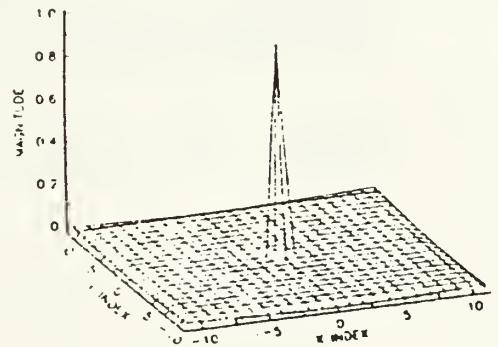
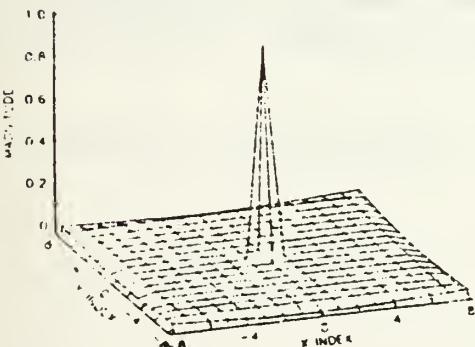
CONVOLUTION OF LAPLACIAN AND ITS  $9 \times 9$  INVERSE



CONVOLUTION OF LAPLACIAN AND ITS  $15 \times 15$  INVERSE



CONVOLUTION OF LAPLACIAN AND ITS  $21 \times 21$  INVERSE



### LIST OF REFERENCES

1. Box, G.E.P., and Jenkins, G.M., Time Series Analysis, Forecasting and Control, Holden-Day, 1976.
2. Therrien, C.W., "Relations Between 2-D and Multichannel Linear Prediction," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-29, No. 3, June 1981.
3. Dudgeon, D.E., and Mersereau, R.M., Multidimensional Digital Signal Processing, Prentice-hall, Inc., 1984.
4. Orfanidis, S.J., Optimum Signal Processing: An Introduction, Macmillan Publishing Co., 1985.
5. Oppenheim, A.V. and Schafer, R.W., Digital Signal Processing, Prentice Hall, Inc., 1975.
6. Peebles, P.Z. Jr., Communications System Principles, Addison-Wesley Publishing Company, Inc., 1976.
7. Cadzoq, J.A., Discrete Time Systems, Prentice-Hall, Inc., 1973.
8. Gonzalez, R.C. and Wintz, P., Digital Image Processing, Addison-Wesley Publishing Company, Inc., 1977.
9. Weber, A.G., "USC SIPI Report 101, Image Data Base," Signal and Image Processing Institute, University of Southern California, February 1986.
10. Brodatz, P., Textures: A Photographic Album for Artists and Designers, Dover, 1966.
11. Modestino, J.W. and Fries, R.W., "Construction and Properties of a Useful Two-Dimensional Random Field," IEEE Transactions, Information Theory, Vol. IT-26, No. 1, January 1980.
12. Beyer, W.H., CRC Standard Mathematical Tables, CRC Press, Inc., 1984.

INITIAL DISTRIBUTION LIST

No. Copies

- |                                                                                                                                                          |   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|---|
| 1. Defense Technical Information Center<br>Cameron Station<br>Alexandria, Virginia 22304-6145                                                            | 2 |
| 2. Library, Code 0142<br>Naval Postgraduate School<br>Monterey, California 93943-5002                                                                    | 2 |
| 3. Department Chairman, Code 62<br>Electrical and Computer Engineering Dept.<br>Naval Postgraduate School<br>Monterey, California 93943-5000             | 1 |
| 4. Professor Charles W. Therrien, Code 62Ti<br>Electrical and Computer Engineering Dept.<br>Naval Postgraduate School<br>Monterey, California 99343-5000 | 3 |
| 5. Professor Roberto Cristi, Code 62Cx<br>Electrical and Computer Engineering Dept.<br>Naval Postgraduate School<br>Monterey, California 93943-5000      | 3 |
| 6. LT Steven C. Rathmanner<br>1453 N. 52nd Ave.<br>Pensacola, Florida 32506                                                                              | 3 |
| 7. Director, Research Administration, Code 012<br>Naval Postgraduate School<br>Monterey, California 93943-5000                                           | 1 |
| 8. Hamdy El-Shaer<br>SMC #1616<br>Naval Postgraduate School<br>Monterey, California 93943-5000                                                           | 1 |















DUDLEY KNOX LIBRARY  
NAVAL POSTGRADUATE SCHOOL  
MONTEREY, CALIFORNIA 93943-5002

Thesis  
R24345 Rathmanner  
c.1      Image texture genera-  
              tion using autoregressive  
              integrated moving average  
              (ARIMA) models.

Thesis  
R24345 Rathmanner  
c.1      Image texture genera-  
              tion using autoregressive  
              integrated moving average  
              (ARIMA) models.

DUDLEY KNOX LIBRARY



3 2768 00032608 6